ASCI μ68™

System X TEN

User's Manual

— 6800 —

ASCI/μ68 System X User's Manual

6800

Copyright ASCI 1979. All rights reserved. No part of this book may be reprinted, or reproduced, or utilized in any form or by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying and recording, or in any information storage and retrieval system, without permission in writing from ASCI Marketing Group.

ASCI MARKETING GROUP
27439 Holiday Lane / Suite 101
Perrysburg, Ohio 43551
Telephone / 419-874-1991

ASCI μ68™

ASCI/LS-UM
6800 System X User's Manual

The μ68 Computer

The μ68 computer and peripherals was designed by ASCI Inc., of Perrysburg, Ohio. The unit was deisgned to allow the novice to quickly and easily learn the fundamentals of microcomputer operation, and to provide the more experienced user with a useful debugging tool. The basic unit is light weight, portable, and exonomical. The μ 68 was designed to allow for future expansion using the Motorola standard EXORCISOR bus compatible peripherals.

The basic unit consists of a two-slot mother board, power supply, hex keyboard, six digit LED display, and CPU card, all housed in an attache sized enclosure. It is an attractive, as well as functional unit.

INDEX

TITLE **PAGE**

I. Power Up/Reset and Display 1

II. ASCI T68 BUG Commands 1-9

III. Stack Area 10

IV. Interrupt Service 10

V. I/O Ports 11

VI. Cassette Tape Interface 11

VII. Loading a Tape 11-12

VIII. Keypad or Terminal Control 12

IX. BASIC 12

X. Baud Rate Selection for System X 13

XI. Jump Vectors 14

XII. Subroutines Parallel Input/Output 15-16

XIII. Serial Input/Output (ACIA) 17-18

XIV. Memory Map 19

XV. Instruction Set 20-21

XVI. Schematics 23-34

XVII. Monitor Listing 35-64

I POWER UP/RESET

System initialization occurs on power up and when RESET is pressed. The system monitor, ASCI T68BUG, configures the I/O for nominal operation. (PIA0 for keyboard, PIA1 for display) and resets all flags and pointers. When initialization is complete, ASCI T68BUG displays ASCI UP and waits for commands to be entered from the keyboard.

DISPLAY

The display consists of six 7-segment Light-Emitting-Diodes (LED). The left-most four digits normally display two byte date (eg., addresses and the index register) while the two digits on the right normally display single byte data (eg., memory contents or accumulator contents).

II ASCI T68BUG COMMANDS

The ASCI T68BUG Monitor supports 16 commands, chosen for their utility in debugging user programs. The various commands are explained below. The procedures to be followed in executing the commands are tabulated following the command descriptions.

The monitor commands are accessed through a hex keypad (shown in Figure 1), with prompts and results displayed on a six-digit seven-segment LED display (also shown on the left). Two additional functions, NMI and RESET are provided through pushbuttons to enable the non-maskable interrupt and hardware reset capabilities of the system.

NOTE: Keyboard has a double function to reduce size.

C Change	**D** Do	**E** Exam	**F** Fwd
8 Punch	**9** Load	**A** Auto	**B** Back
4 Pc/cc	**5** X/A	**6** Sp/B	**7** Set Pc
0 Jump	**1** Branch	**2** SS	**3** BP

Figure 1. Keyboard/Display

SET PC (Set User Program Counter)

This command allows the user to set the Program Counter. The instruction pointed to by this vector will be the first one executed prior to a DO, SINGLE STEP, or BREAK-POINT command. Failure to execute this command prior to the initial DO, SINGLE STEP, or BREAK-POINT commands will result in unpredictable system operation.

SINGLE STEP (SS)

This command allows the user to execute a program in RAM one instruction at a time. After executing an instruction, the program counter and condition code register are displayed. Control then reverts to ASCI T68BUG, and all monitor commands are available again.

BREAK-POINT (BP)

This command allows the user to execute code in RAM until a certain specific address (the break point) is encountered. When the PC reaches the break point, the PC and CC are displayed, and control reverts to ASCI T68BUG.

PC/CC (Display Program Counter and Condition Code)

This command allows the user to display the current Program Counter and Condition Code register.

X/A (Display Index Register and Accumulator A)

This command allows the user to display the current Index Register and Accumulator A.

SP/B (Display Stack Pointer and Accumulator B)

This command allows the user to display the current Stack Pointer and Accumulator B.

PUNCH

This command, when used with the optional tape recorder, allows the user to dump object code from RAM to tape. The code is stored on tape in the format required by the LOAD command.

LOAD

This command, when used with the optional tape recorder, allows the user to load object code from tape into RAM. The code must have previously been stored on tape using the PUNCH command. Error checking is done during the LOAD, and an unsuccessful load will result in an error code being displayed.

AUTO STEP

This command allows the user to enter data into sequential RAM locations after specifying an initial address. Exit from this mode is by RESET only.

BACKWARD

This command, active only while in EXAMINE mode, allows the user to inspect the contents of a previous memory location.

CHANGE

This command, active only in EXAMINE mode, allows the user to modify the contents of the location currently being displayed by entering the new 2-digit hex contents of the location.

DO

This command causes unconditional execution of the user program, beginning with the instruction pointed to by the current Program Counter.

EXAMINE

This command allows the user to examine the contents of any location in memory space. After entering this mode, the CHANGE, BACKWARD, and FORWARD commands become active.

FORWARD

This command, active only in the EXAMINE mode, allows the user to inspect the contents of the next sequential memory location.

JUMP (Jump to Address)

This command allows the user to examine the location in memory identified by the byte currently being displayed and the previous byte. The byte in the display is the LSB and the previous byte is the MSB. This facilitates debugging by allowing the user to simulate the execution of jump instruction in the Examine Memory Mode.

BRANCH (Branch to Offset)

This command allows the user to examine the location in memory identified by the byte in display, that byte treated as the offset of a branch instruction. This facilitates debugging by allowing the user to simulate the execution of branch instructions in the Examine Memory Mode to verify that the destination of the branch is correct.

NMI

This pushbutton generates a non-maskable interrupt. So long as the user has not modified the NMI vector, the ASCI T68BUG monitor service the NMI by performing a PC/CC command, then waiting for further instructions. This provides a method for halting a runaway user program (so long as that program has not changed the NMI vector).

RESET

This pushbutton generates a hardware reset. When activated, the monitor is initialized. This pushbutton provides the highest level interrupt capability. If a program is completely out of control, and NMI doesn't return control to the monitor, RESET will.

ASCI T68BUG Command Procedures

Procedure:	Press	Enter	Display	Monitor Action
1	SET PC		- - - - PC	
		DDDD	DDDD PC	DDDD Stored as new program counter.
2	SINGLE STEP		PPPP CC	The instruction pointed to by the user PC will be executed. Control then returns to ASCI T68BUG and the PC/CC command is automatically executed.
3	BREAK POINT		- - - - bP	
		DDDD	DDDD bP	The instruction at DDDD will be saved, then replaced with a SWI ($3F). A GO command will then be executed automatically. When the SWI command is encountered, control reverts to ASCI T68BUG and the PC/CC command is automatically executed.
			PPPP CC	
4	PC/CC		PPPP CC	The current user Program Counter and Condition Code Register are displayed.
5	X/A		XXXX AA	The current user Index Register and Accumulator A are displayed.
7	PUNCH		- - - - PU	Dump contents of memory starting with bA (beginning address) and continuing to EA (ending address) to Tape in proper format to be loaded using the LOAD Command.
		DDDD	- - - - bA	
		DDDD	DDDD EA	

8	LOAD		LOAD - -	Loads program into RAM from Tape. Program must previously have been stored using PUNCH command. Displays 00 if load is successful, 01 if error occurs.
			LOAD 00	
			LOAD 01	
9	AUTO STEP		- - - - - -	Enter automatic entry mode. ASCI T68 BUG opens sequential memory locations beginning with DDDD, and allows you to enter new contents. Exit mode by pressing RESET.
		DDDD	DDDD - -	
		DD		
10	BACKWARD		DDDD DD	When in EXAMINE mode, location counter is decremented and current contents of the location are displayed.
11	CHANGE		DDDD - -	When in EXAMINE mode, the current location is opened (- - displayed).
		DD	DDDD DD	New contents are then entered (DD).
12	DO			User program executes from the current location of the PC.
13	EXAMINE		- - - - - -	Enter EXAMINE mode. Display current contents of location DDDD.
		DDDD	DDDD XX	
14	FORWARD		DDDD XX	When in EXAMINE mode, location counter is incremented and current contents of the location are displayed.

15	NMI	PPPP CC	Non-maskable interrupt, resulting in display of PC and CC when NMI occured. All ASCI T68BUG commands are then available.
16	RESET	ASCI UP	Hardware reset. All variables and I/O initialized.
17	JUMP	DDDD DD	When in EXAMINE mode, the contents of the memory location specified by the currently displayed data (at the LSB) and the preceding location data (as the MSB) is displayed.
18	BRANCH	DDDD DD	When in EXAMINE mode, the contents of the memory location obtain by adding the currently display data to the currently displayed address as a branch offset is displayed.

III STACK AREA

The ASCI T68BUG monitor maintains separate stack areas for user and monitor operation. On power up (or RESET) a ten byte area of RAM ($0000-$0009) is allocated for the users stack (**Note:** This area is sufficient to run and debug user programs with one level of subroutines only.) Normal programming practice would require the user program to initialize its own stack area.

ASCI T68BUG maintains a separate working stack so that monitor operations do not affect the user stack area. Execution of an SWI or NMI while a user program is running will result in the registers being saved on the user stack. The user stack pointer is stored at $000C and can be examined using the EXAMINE or SP/B command keys.

IV Interrupt Service

The ASCI T68BUG monitor supports user IRQ (Hardware), NMI (Non-maskable), and SWI (Software) interrupts. The vectors to the service routines are stored and initialized at POWER UP/RESET as follows:

Interrupt	Address of Vector	Initialized to T68BUG
IRQ	$0022	ENTADR
SWI	$0024	PCCCEX
NMI	$0026	NMISRV

A user program may change these vectors to point to specialized application routines while maintaining most of the monitor functions. (Only the NMI feature is inhibited. The ASCI T68BUG monitor uses SWI to implement BREAKPOINT and SINGLE STEP. These functions are maintained even when SWIADR points to a user routine by having the SWI vectored first to the ASCI T68BUG service routine, and then to the user SWIADR only if the SWI was not initiated by ASCI T68BUG.)

V I/O Ports

The μ68 has three parallel and one serial port. The parallel ports, PIA0 and PIA1, are dedicated to the keyboard and display respectively. The third parallel port is available to the user. (**Note:** PIA0 and PIA1 are also available to the user at the card edge if the keyboard and display are not required.) The serial port, ACIA, is dedicated to the cassette interface is not required. (See the circuit and layout drawings for information on card edge pin assignments).

VI Cassette/Paper Tape Interface

The μ68 contains an RS-232 interface that is used by ASCI T68BUG to access the cassette recorder or TTY paper tape reader/punch. The data is punched/read utilizing the standard Morotola EXBUG format.

VII LOADING A TAPE

When the LOAD command occurs in command mode, T68BUG initialized the ACIA and sends the reader on character (DC1, $11) to the peripheral device. It then monitors the incoming data, watching for the segment tag. 'S'. If the byte following the 'S' is not a '1' or a '9', ASCI T68BUG will display 'BAD', then reads off character (DC3, $13), and return to monitoring the keyboard for a new command.

If the 'S1' combination is received, the frame (two characters), is interpreted as the byte count for this segment. The next two frames give the base address for the remainder of the segment. The data from succesive frames in the segment is loaded into sequential memory locations starting at the base address. The last byte in the segment is the checksum over the segment. The transmitted checksum must equal the computed checksum, if not, the T68BUG will display 'BAD', send the reader off character, and return to monitoring the keyboard. If the checksums are equal, T68BUG watches for the next segment tag 'S', followed by '1' or '9'.

If the tag is 'S9', ASCI T68BUG sends the reader off character, displays 'GOOD', and returns to monitoring the keyboard.

VIII KEYPAD OR TERMINAL CONTROL

The μ 68 Trainer supports both keypad and terminal control. The source for all commands is determined on Power Up/Reset by the first key pressed - if the key comes from the keypad, subsequent commands may be entered from it; if an "**A**" is typed on the terminal, subsequent commands may be entered from it.

The only active keys on the terminal are 0 - 9 and A - F and these keys have the same functions as the corresponding keys on the keypad. The LED display is **not** affected by the control mode.

If, during a session, it is desired to change control modes without doing a RESET, the MSB of IOFLAG (Address = $50) may be changed. When this bit is 0, the keypad has control; when it is 1, the terminal has control.

IX BASIC

To load BASIC, set the PC = $F836 and press D (DO). The BASIC loader checks to see that the control mode = terminal, and then checks to see if the BASIC PROMS have been installed properly. If both conditions are met, BASIC is moved to RAM and control is transferred there.

X BAUD RATE SELECTION FOR SYSTEM X

Depress Examine Key, address 0043. It will read FF which is 110 baud. Depress Change and change to desired baud rate as stated below.

110 Baud = FF
150 Baud = FE
300 Baud = FD
600 Baud = F6
1200 Baud = FB
1800 Baud = FA
2400 Baud = FC
4800 Baud = F9
9600 Baud = F8

Please note the Baud Rate Generator Memory Location 0043 will reset to FF anytime the system is RESET.

AUTOMATIC BAUD RATE SELECTION

If the μ 68 System X is placed under terminal control. The baud rate generator address 0043 will be set automatically by the leading edge of the first square wave. As the result the System X baud rate will be the baud rate of the terminal controlling.

XI JUMP VECTORS

For easy accessibility to subroutines the ASCI System X has jump vectors established for those subroutines most often used.

To use these jump vectors, use the JMP instruction and address the subroutine you desire to use JUMP VECTOR.

EX. To go to the KEYIN subroutine program as follows:

7E (JMP) F803 (JUMP VECTOR)

The following list will give the most commonly used subroutines, labels, and their JUMP VECTORS. For detailed explanations of these subroutines, see the following pages, for their respective programs, see the printout of the ASCI T68BUG

Subroutine Label	Jump Vector
DISPLY	F80C
WRDATA	F80F
KEYIN	F803
OUT2HX	No Jump Vector
ADROUT	F812
IN2HX	F806
INEE	F81B
INHEX	F81E
IN2	F821
IN4	F824
OUTEE	F827
OUTHEX	F82A
OUT2	F82D
OUT4	F830
BAUD	F833

For additional subroutine locations, see page 4 of ASCI T68BUG Printout.

XII Subroutines Parallel Input/Output

The **DISPLY** Subroutine Memory location FD74 Jump Vector F80C

The **DISPLY** subroutine displays 6 characters in a single left-to-right scan of the six LED's of the ASCI System X. The 6 characters displayed had been stored earlier as bit patterns which cause the appropriate 7 segment characters to light up. The 6 bit patterns for the **DISPLY** subroutine are fetched from locations C04A, 004b, 004C, 004d, 004E, and 004F. The **WRDATA** routine may be used to accomplish the number-to-character bit Pattern Conversion.

The **WRDATA** Subroutine Memory location FE22 Jump Vector F80F

The **WRDATA** subroutine converts a 4-bit binary number, passed via the A accumulator into a bit pattern which lights up the corresponding hexadecimal character in a LED display. The subroutine uses a sequential table of patterns found at location FD26 through FD35 which represent the hexadecimal characters 0 through F, respectively. The LOAD DISPLAY subroutine can be disabled by setting the DISPLAY INHIBIT flag (NOWRF) at memory location 0044 to a "1".

The **KEYIN** Subroutine Memory location FD52 Jump Vector F803

The **KEYIN** subroutine scans the 16 keys of the keyboard for input and calls on the **DISPLY** subroutine to refresh the LED display between each scan. The keys have sequential priority and the "F" key has the highest priority.

The **KEYIN** subroutine uses the **WRDATA** subroutine to save the character input from the keyboard. **KEYIN** continues to loop (always calling **DISPLY** after each scan) until depressed key is released. The character will be displayed unless the display inhibit flag (memory location 0044) is set.

The **OUT2HX** subroutine Memory location FD27 No jump vector

The **OUT2HX** subroutine converts an 8-bit binary number passed via the A accumulator into two bit patterns. These patterns light up two corresponding hexadecimal characters in the LED display. The 8-bit binary number is split into two 4-bit numbers. The **WRDATA** subroutine is called twice in order to write the two display patterns in two sequential effective memory locations. The 8-bit data word is life in the B accumulator upon exit from the subroutine.

The **ADROUT** subroutine Memory location FD1B Jump vector F812

The **ADROUT** subroutine converts a 16-bit number to 4 LED display codes. The 16-bit number is fetched by the **ADROUT** subroutine from locations 0038 (8-bits) and 0039 (8-bits). The **OUT2HX** subroutine is called twice, once for each 8-bit byte. The 7-segment LED codes are placed in locations 004A-004F and will display in the first 4 LED's.

The **IN2HEX** subroutine Memory location FD38 Jump vector F806

The **IN2HEX** subroutine builds a composite 8-bit binary number from two keyboard entries. It makes two calls on the **KEYIN** subroutine. The first user entry at the keyboard becomes the upper 4-bits and the second keyboard entry becomes the lower 4-bits of the final 8-bit binary number. The A accumulator contains this 8-bit number upon exit from the subroutine.

XIII Serial Input/Output (ACIA)

The **INEE** subroutine Memory location FC46 Jump vector F818

The **INEE** subroutine gets one character from the ACIA and leaves that character in the A accumulator upon exit from the subroutine.

The **INHEX** subroutine Memory location FC33 Jump vector F81E

The **INHEX** subroutine gets one ASCI II character from the ASIA by use of the INEE subroutine converts the ASCI II character to hexadecimal and leaves conversion in the A accumulator upon exit from the subroutine.

The **IN2** subroutine Memory location FC22 Jump vector F821

The **IN2** subroutine gets two ASCI II characters from the ACIA and uses the **INHEX** subroutines twice to convert the ASCI II character to hexadecimal. This subroutine uses the byte counter (address 0042) and leaves the conversion in the user stack area (Memory locations 0000 through 000A.)

The **IN4** subroutine Memory location FC20 Jump vector F824

The **IN4** subroutine gets 4 ASCI II characters or 4 hexadecimal characters from the ACIA and uses the **IN2** and **INHEX** subroutines.

The **OUTEE** subroutine Memory location FC6E Jump vector F827

The **OUTEE** subroutine takes the byte in the A accumulator and sends that byte to the ACIA.

The **OUTHEX** subroutine Memory location FC66 Jump vector F82A

The **OUTHEX** subroutine sends the hexadecimal digit in the A accumulator converts that digit to ASCI II and send it to the ACIA. The **OUTHEX** subroutine uses the **OUTEE** subroutine and the **DISPLY** subroutine.

The **OUT2** subroutine Memory location FC55 Jump vector F82D

The **OUT2** subroutine takes 2 hexadecimal characters from memory via the X register, converts these characters to ASCI II by use of the **OUTHEX** subroutine and sends them to the ACIA.

The **OUT4** subroutine Memory location FC53 Jump vector F830

The **OUT4** subroutine through the use of the **OUT2** subroutine twice, take 4 hexadecimal characters in memory, converts these characters to ASCI II and sends them to the ACIA.

The **BAUD** subroutines initialize the ACIA (Memory Address EC00) and sets the **BAUD** Rate to 110. Memory location 0043 is the Baud Rate Code and this may be changed for higher Baud Rate operation.

XIV MEMORY MAP

The ASCI System X has 1K bytes of RAM (Random Access Memory) and provision for 2K or 4K bytes of PROM (Programmable Read Only Memory). The basic unit has the ASCI T68BUG monitor in this PROM space and utilized 64 bytes of RAM for data storage. The remainder of RAM is available for the users programs and/or data storage.

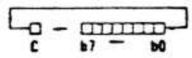
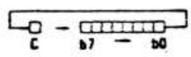
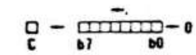
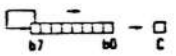
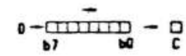
ADDRESS

FFFF	ASCI TBUG	2708 PROM 2
F800	2716 PROM 2	2708 PROM 1
F7FF F000	2716 PROM 1	
EC00	ACIA	
E800	Spare PIA	
E400	Display PIA	
E000	Keyboard PIA	
DFFF 0400	Expansion Memory Space	
03FF 0060	User RAM	
005F 0000	ASCI T68BUG RAM	

Figure - Memory Map

XV Instruction Set

TABLE 3 - ACCUMULATOR AND MEMORY INSTRUCTIONS

		ADDRESSING MODES															BOOLEAN/ARITHMETIC OPERATION	COND CODE REG.					
		IMMED			DIRECT			INDEX			EXTND			IMPLIED			(All register labels refer to contents)	5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	=	OP	~	=	OP	~	=	OP	~	=	OP	~	=		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				$\overline{M}$ → M	•	•	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	•	•	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	•	•	↕	↕	R	S
Complement, 2's	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊙M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊙M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A 	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A 	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A 	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A 	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A 	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B	•	•	R	↕	⑥	↕
Store Acmltr	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

LEGEND:

OP	Operation Code (Hexadecimal);
~	Number of MPU Cycles;
=	Number of Program Bytes;
+	Arithmetic Plus;
−	Arithmetic Minus;
·	Boolean AND;
M_{SP}	Contents of memory location pointed to be Stack Pointer;
+	Boolean Inclusive OR;
⊙	Boolean Exclusive OR;
■	Complement of M;
→	Transfer Into;
0	Bit = Zero;
00	Byte = Zero;

Note – Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

H	Half carry from bit 3;
I	Interrupt mask
N	Negative (sign bit)
Z	Zero (byte)
V	Overflow, 2's complement
C	Carry from bit 7
R	Reset Always
S	Set Always
↕	Test and set if true, cleared otherwise
•	Not Affected

TABLE 4 — INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

		IMMED			DIRECT			INDEX			EXTND			IMPLIED				COND. CODE REG.					
																		5	4	3	2	1	0
POINTER OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BOOLEAN/ARITHMETIC OPERATION	H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M + 1)$	•	•	⑦	↕	⑧	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	•	↕	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	•	↕	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H, (M + 1) \rightarrow X_L$	•	•	⑨	↕	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H, (M + 1) \rightarrow SP_L$	•	•	⑨	↕	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M, X_L \rightarrow (M + 1)$	•	•	⑨	↕	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M, SP_L \rightarrow (M + 1)$	•	•	⑨	↕	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

TABLE 5 — JUMP AND BRANCH INSTRUCTIONS

		RELATIVE			INDEX			EXTND			IMPLIED				COND. CODE REG.					
															5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	OP	~	#	OP	~	#	OP	~	#	BRANCH TEST	H	I	N	Z	V	C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	•
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	•
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	•	•	•
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	•	•	•	•	•	•
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	•	•	•	•	•	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	•	•	•	•	•	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	•	•	•	•	•	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advances Prog. Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1		•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	See Special Operations	•	•	•	•	•	•
Wait for Interrupt*	WAI										3E	9	1		•	⑪	•	•	•	•

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

TABLE 6 — CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

		IMPLIED				COND. CODE REG.					
						5	4	3	2	1	0
OPERATIONS	MNEMONIC	OP	~	#	BOOLEAN OPERATION	H	I	N	Z	V	C
Clear Carry	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	⑫					
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

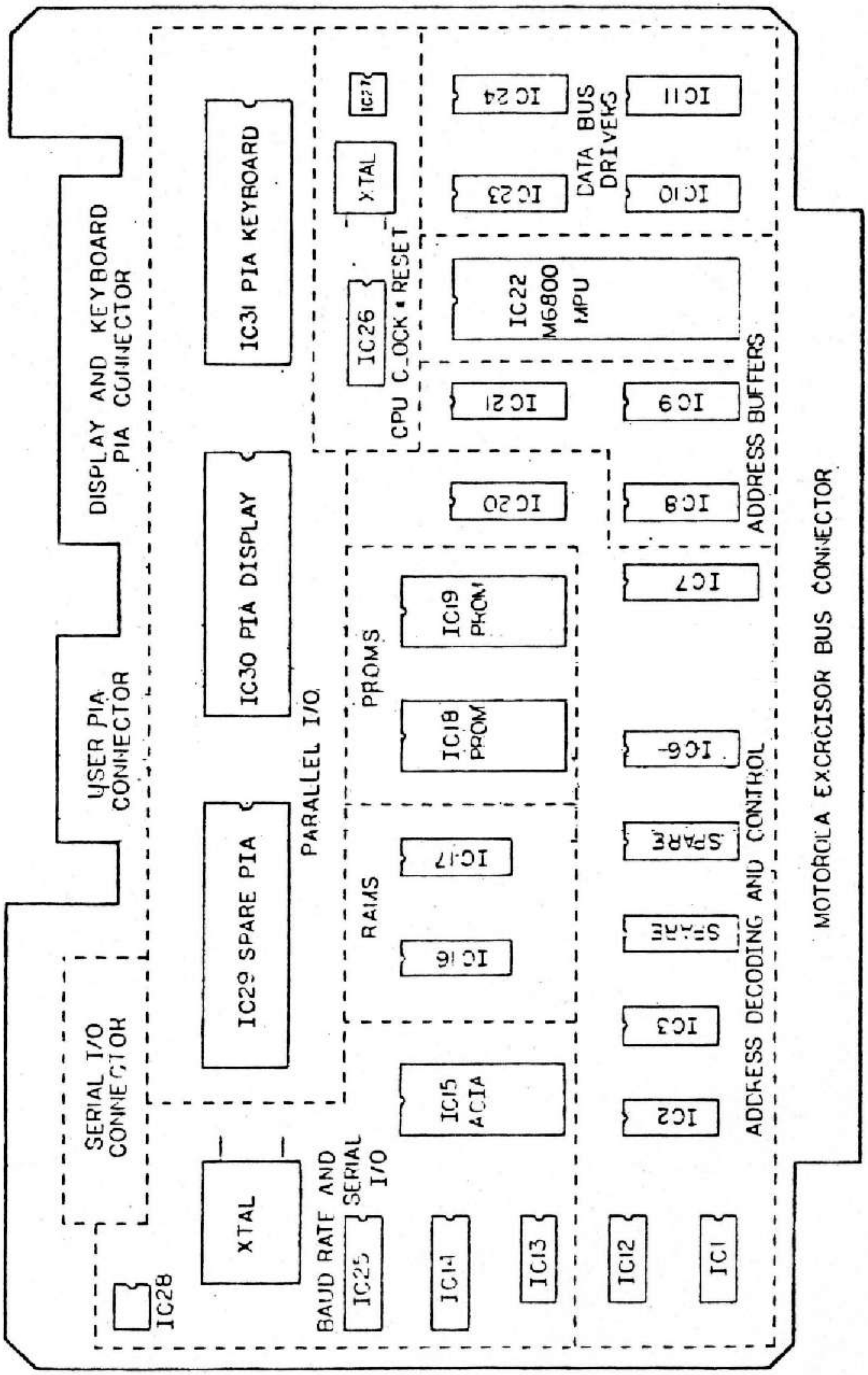
CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

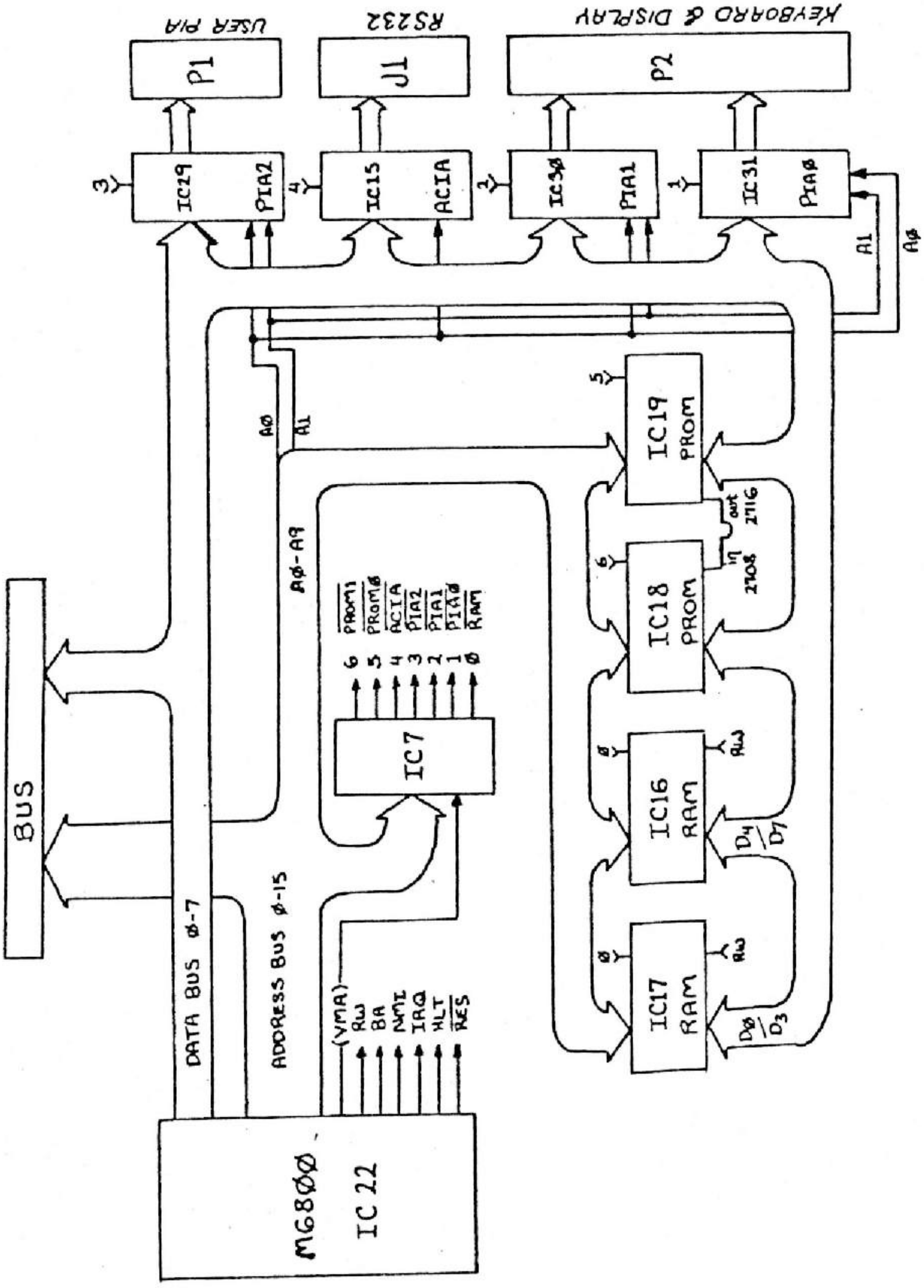
1	(Bit V)	Test: Result = 10000000?
2	(Bit C)	Test: Result ≠ 00000000?
3	(Bit C)	Test: Decimal value of most significant BCD Character greater than nine? (Not cleared if previously set.)
4	(Bit V)	Test: Operand = 10000000 prior to execution?
5	(Bit V)	Test: Operand = 01111111 prior to execution?
6	(Bit V)	Test: Set equal to result of N⊕C after shift has occurred.
7	(Bit N)	Test: Sign bit of most significant (MS) byte = 1?
8	(Bit V)	Test: 2's complement overflow from subtraction of MS bytes?
9	(Bit N)	Test: Result less than zero? (Bit 15 = 1)
10	(All)	Load Condition Code Register from Stack. (See Special Operations)
11	(Bit I)	Set when interrupt occurs. If previously set, a Non-Maskable interrupt is required to exit the wait state.
12	(All)	Set according to the contents of Accumulator A.

ASCI/U68 SYSTEM X CPU PARTS LIST

DESIGNATIONS	DESCRIPTION	PART NUMBER
IC1	Invertor	74LS04
IC2	Dual Input Nand Gates	74LS00
IC3	Three Input Nand Gates	74LS10
IC4, IC5	Spare	
IC6, IC8, IC9, IC21	Amplifier Non-inverting	74LS367
IC7	Address De-coder	74S471
IC10, IC11, IC23, IC24	Bus Drivers, Three State	8T26
IC12	Amplifier Inverting	7406
IC13	Amplifier Inverting	75189
IC14	Control Amplifier	74LS175
IC15	Asynchronous Communications Interface Adapter (ACIA)	MC6820
IC16, IC17	1K Bytes by 4-bit RAMS	2114
IC18, IC19	EPROMS Selectable for 2K-bytes or 4K-bytes	2616 or 2608
IC20	Amplifier Inverting	74LS368
IC22	MPU, Microprocessor Unit	MC6800
IC25	Baud Rate Generator	34702
IC26	Two-Phase Microprocessor Clock	MC6875
IC27	Timing Buffer for RESET	555
IC28	Dual Input And Gates with Inverting Amplifiers	75150
IC29, IC30, IC31	PIA (Peripheral Interface Adapter)	MC6821



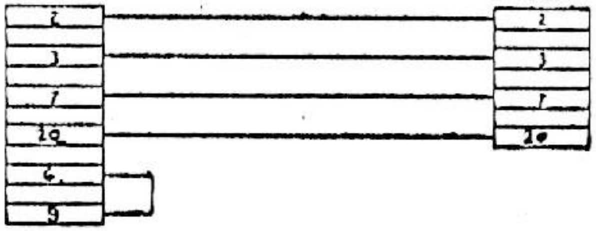
ASCI / µ68 SYSTEM X CPU



ASCI SYSTEM X SYSTEM CONFIGURATION

PARTS LIST

DESIGNATION	DESCRIPTION	QT
CONNECTOR	205208-1	1
PIN	25 PINS · 205089-1	1
HOOD	204178-1	1
CABLE	7 COND. 9410 BELDEN	6'



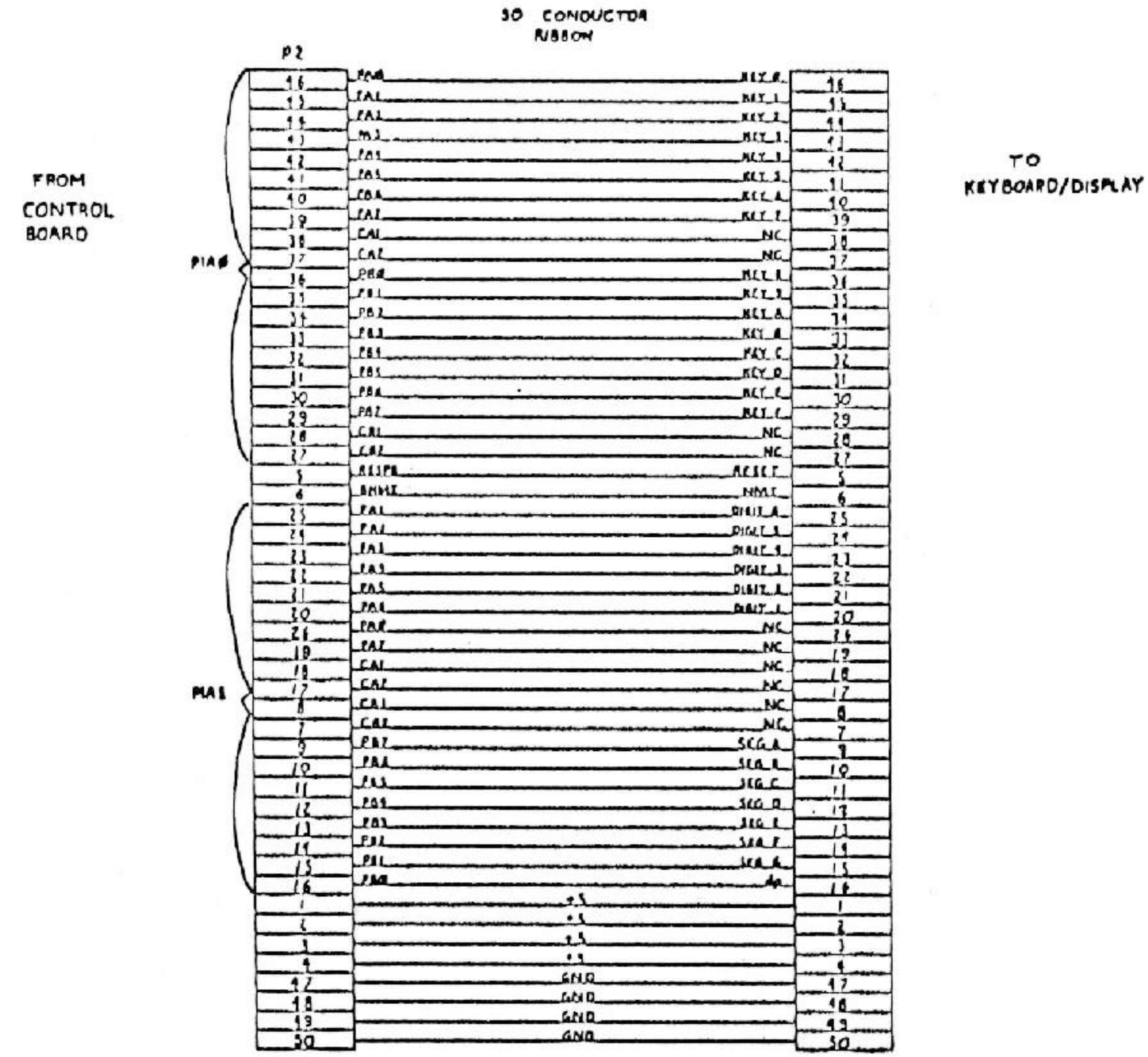
ASCI/μ68

7/5/79

ASCI RS-232 CABLE

PART LIST

DESIGNATION	DESCRIPTION	QTY
CONNECTOR	609-5015 M	2
CABLE	50 COND 171-50 ANSLEY	1x

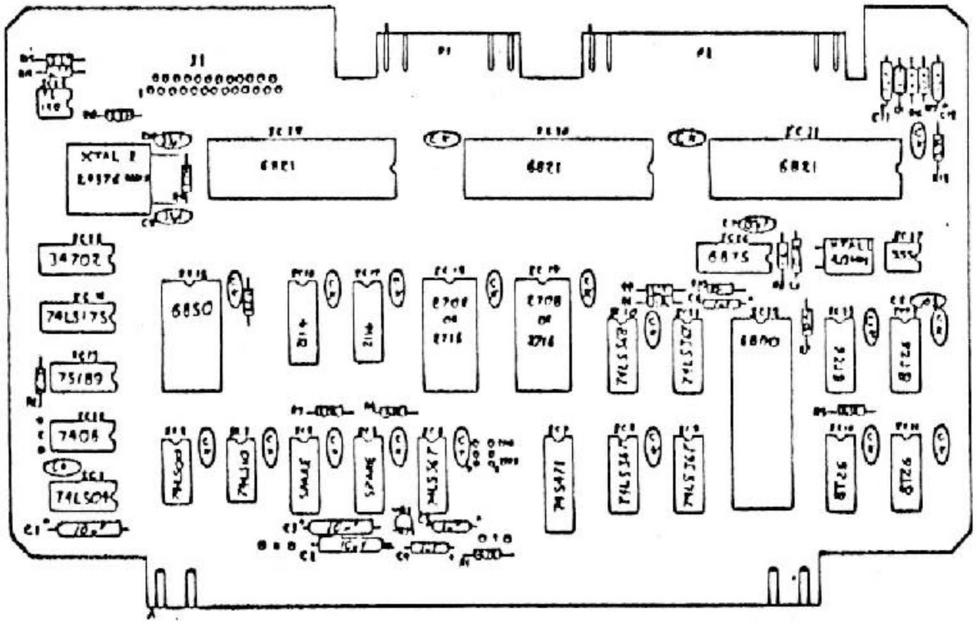


ASCI/U68

DATE 7/5/79 DJI

ASCI KEYBOARD CABLE

577



PART LIST

DESIGNATION	DESCRIPTION	QTY
IC1	74LS04	
IC2	74LS00	1
IC3	74LS10	
IC4, IC5	SPARE	
IC6, IC8, IC9, IC21	74LS367	
IC7	74S471	1
IC10, IC11, IC23, IC24	8T26	4
IC12	7406	
IC13	75189	
IC14	74LS173	
IC15	6850	1
IC16, IC17	2114	2
IC18, IC19	2708 OR 2716	2
IC20	74LS368	1
IC22	MC6800	
IC25	34702	1
IC26	6875	1
IC27	555	1
IC28	75150	1
IC29, IC30 IC31	6821	3
D1	IN4004	1
L1	10uH	1
C1, C2, C3	10uF	3
C4, C5, C6, C11, C12	1uF	5
C7	150pF	
C8	.05uF	
C9, C10	33pF	
C *	.01uF BY PASS CAP.	20
R1	4.7K	1
R2, R4, R14, 15, R18	2.2K	
R3	330Ω	1
R5, R6, R8, R9, R13	3.3K	
R7, R10	10Ω	2
R11	1K	1
R12	10M	1
R16, R17	470K	2
VR1	79L05	1
XTAL 1	4.0MHz	1
XTAL 2	2.4576 MHz	1

ASCI/U68

SCALE:
DATE: 4/30/79
APPROVED BY: [illegible]
DRAWN BY AH
REVISED

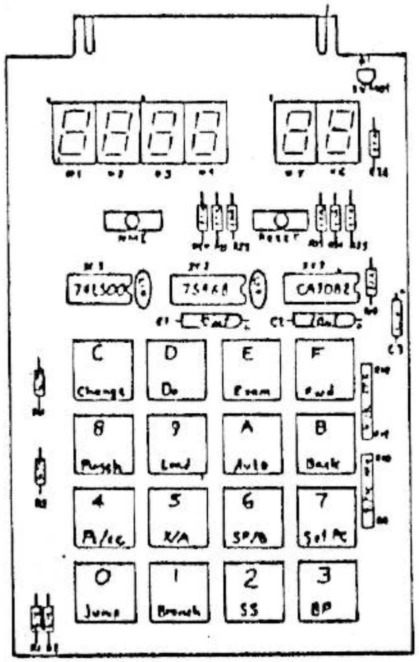
ASCI U68

COMPONENT LAYOUT

DRAWING NUMBER 547

PAR LIST

DESIGNATION	DESCRIP . ON	Q
IC 1	74LS00	1
IC 2	75468	1
IC 3	CA3082	1
Q1	2N4401	1
R1-R8, R10	4.7K	
R9, R11	47K RESISTOR PACKAGE	
R19 - R26	47Ω .	
C1, C2	10µF	2
C3	1µF	1
(C-)	.01µF BYPASS CAPACITORS	
DISPLAY	MAN 74A	
PB SWITCHES	#B121 (C&K)	

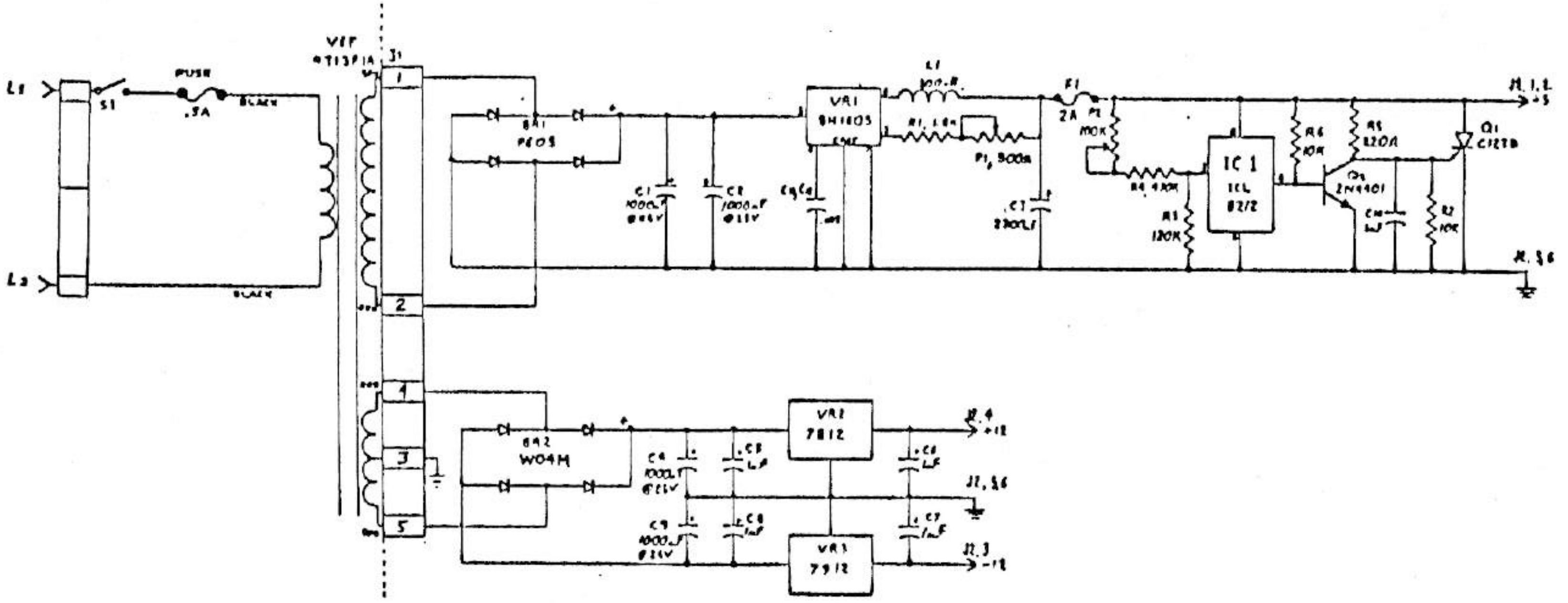


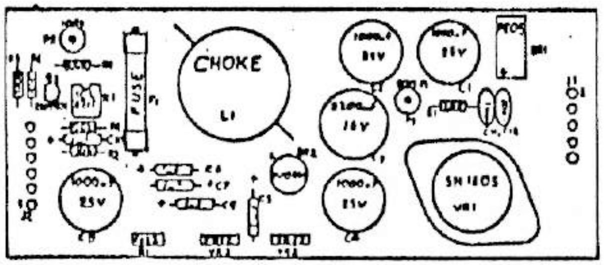
ASCI/µ68

DATE 4/24/79

ASCI µ68 KEYBD /DISPLAY REV 1

COMPONENT LAYOUT 557





PARTS LIST

DESIGNATION	DESCRIPTION	Q
IC1	ICL 8212	
R1	1.8K	1
R2, R6	10K	2
R3	120K	
R4	470K	
R5	220 Ω	
P1	500 Ω POT	
P2	100K POT	1
C1, C2, C4, C9	1000µF 25V	4
C3	2200µF 16V	
C5, C6, C7, C8, C10	1µF	
C12	.002µF = .001+1	
F1	2A	1
Q1	C122B SCR	1
Q2	2N4401	
L1 (CHOKE)	300µH	
BR1	PE05	
BR2	W04M	
VR1	SH1105	1
VR2	7812	1
VR3	7912	1

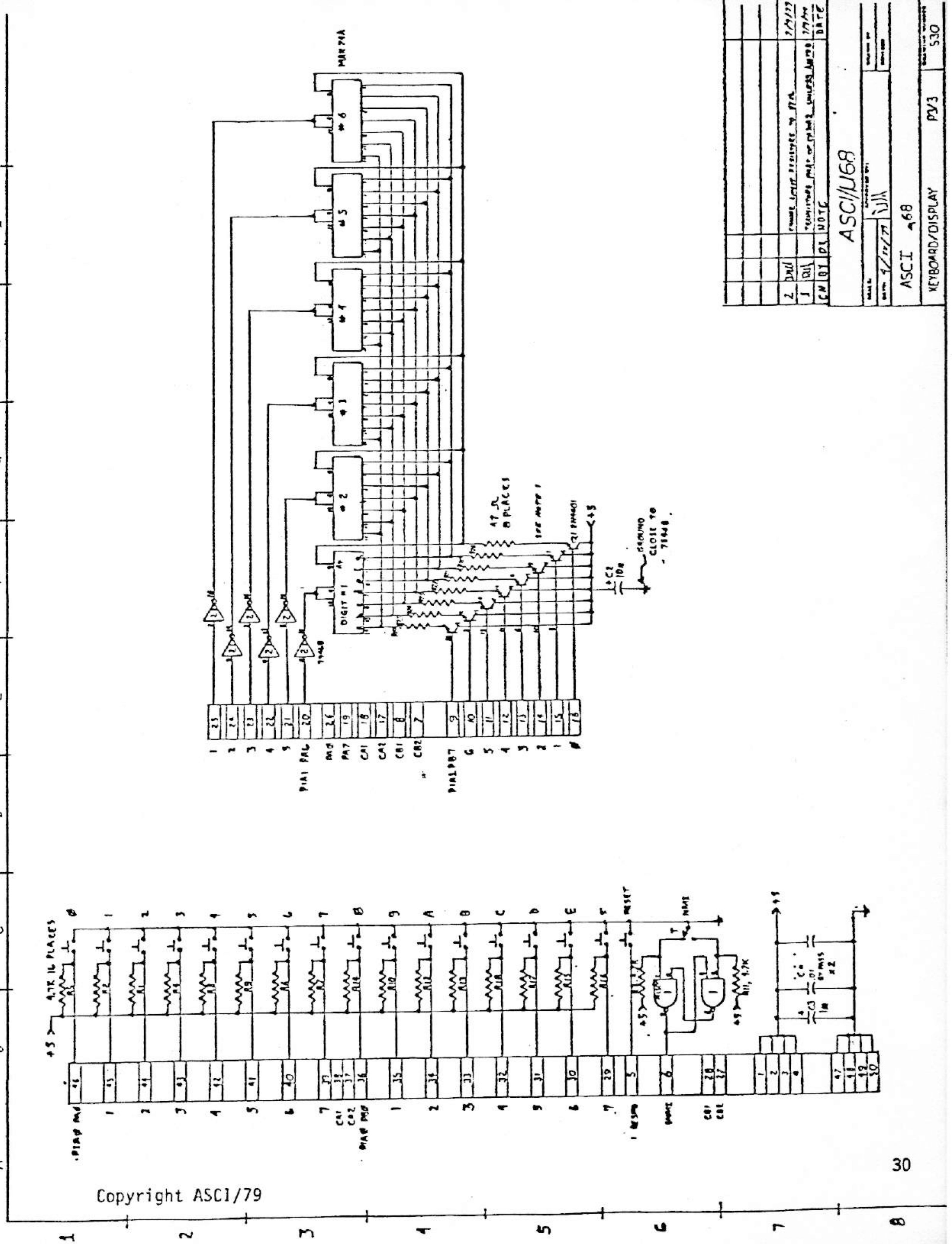
ASCI/U68

DATE 5/11/79

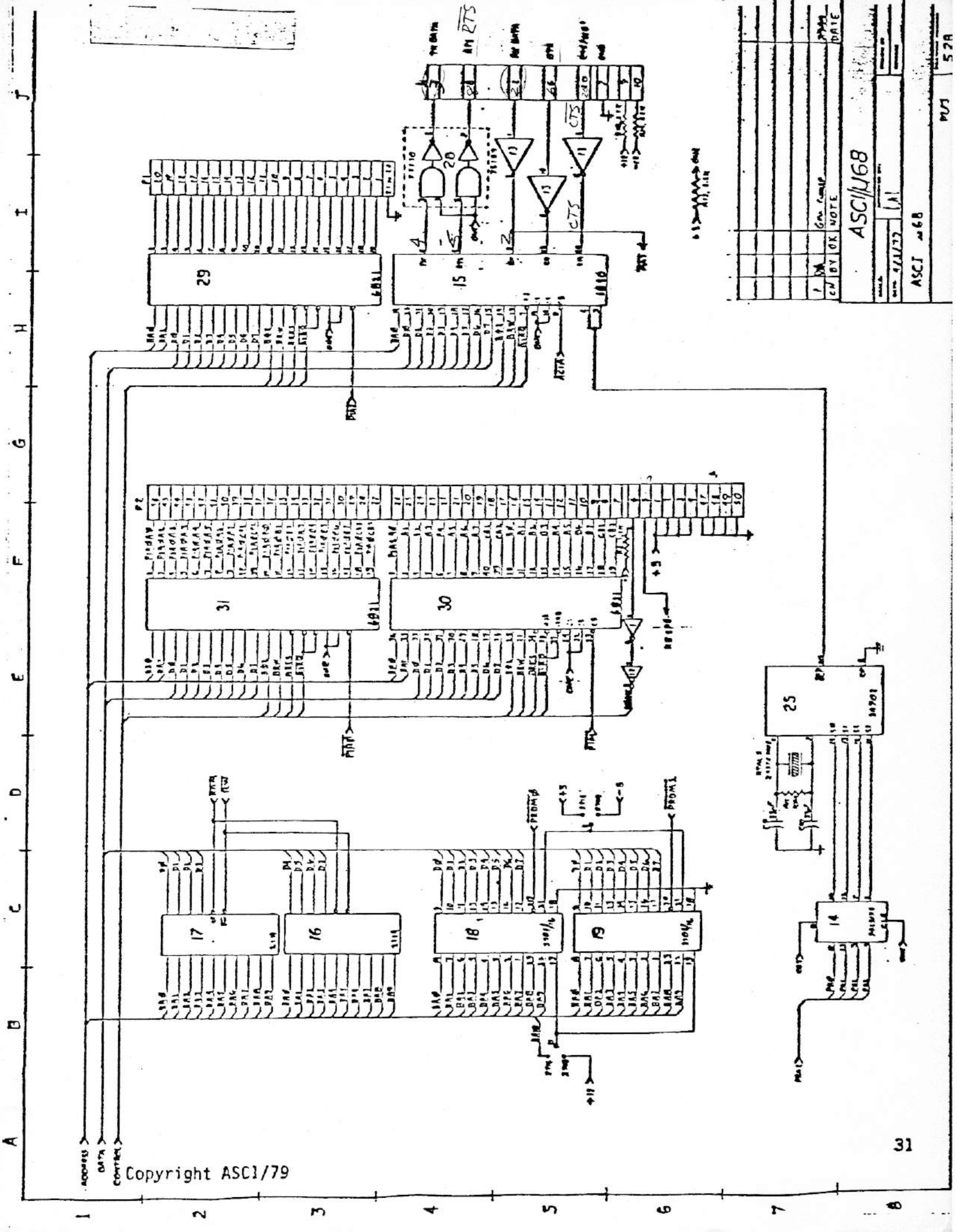
DRAWN BY MR

POWER SUPPLY ASCI

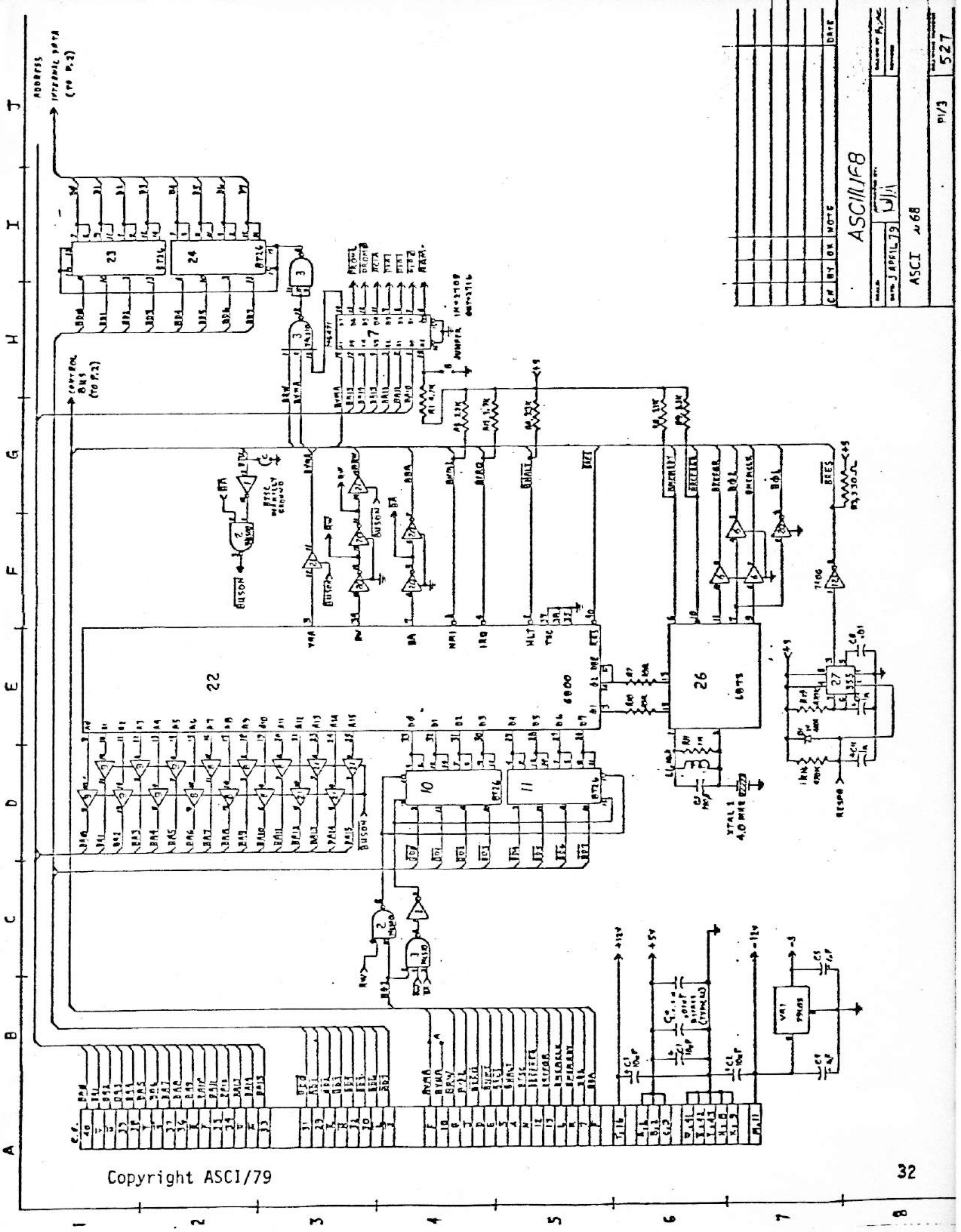
AND COMPONENT LAYOUT



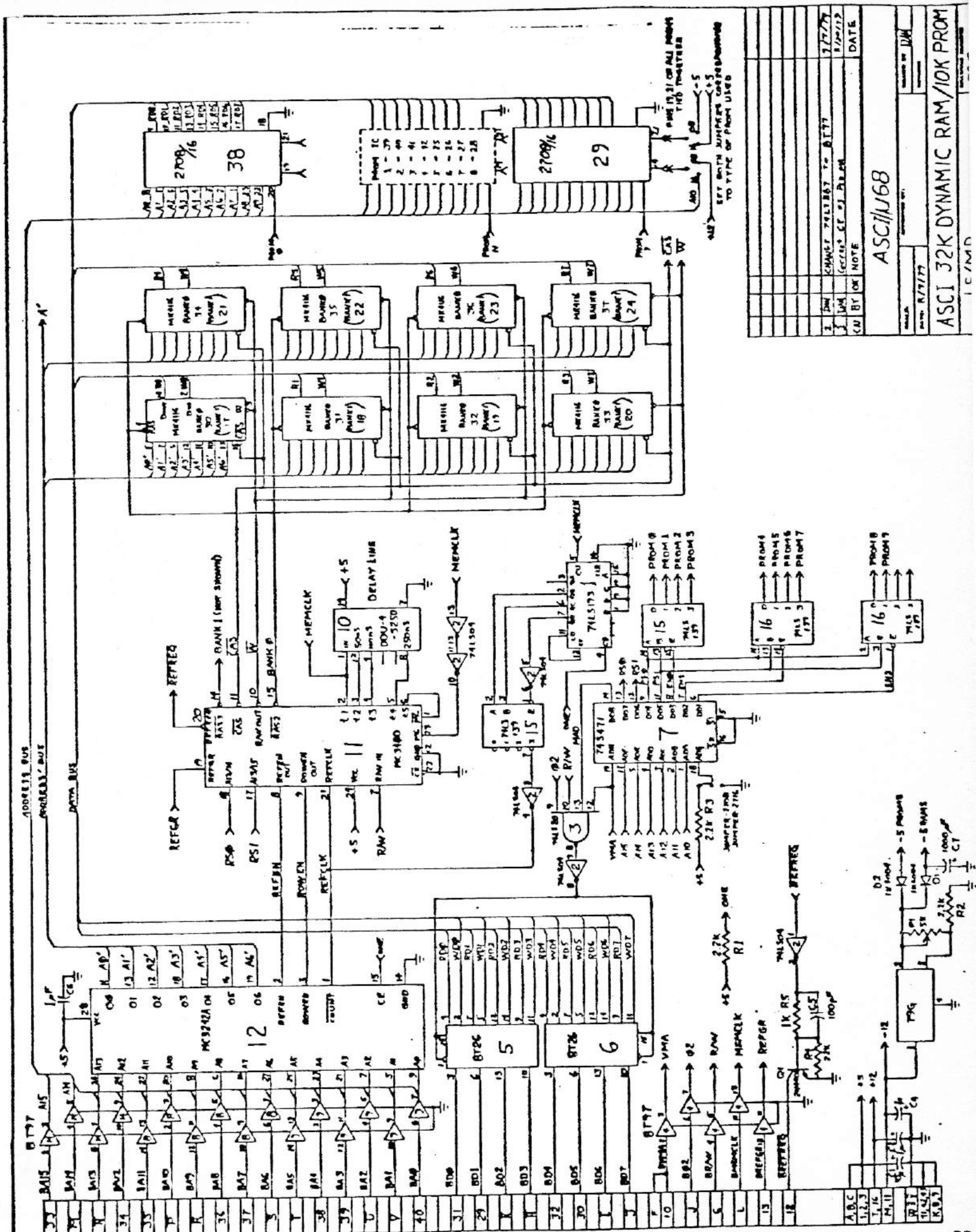
Copyright ASCI/79



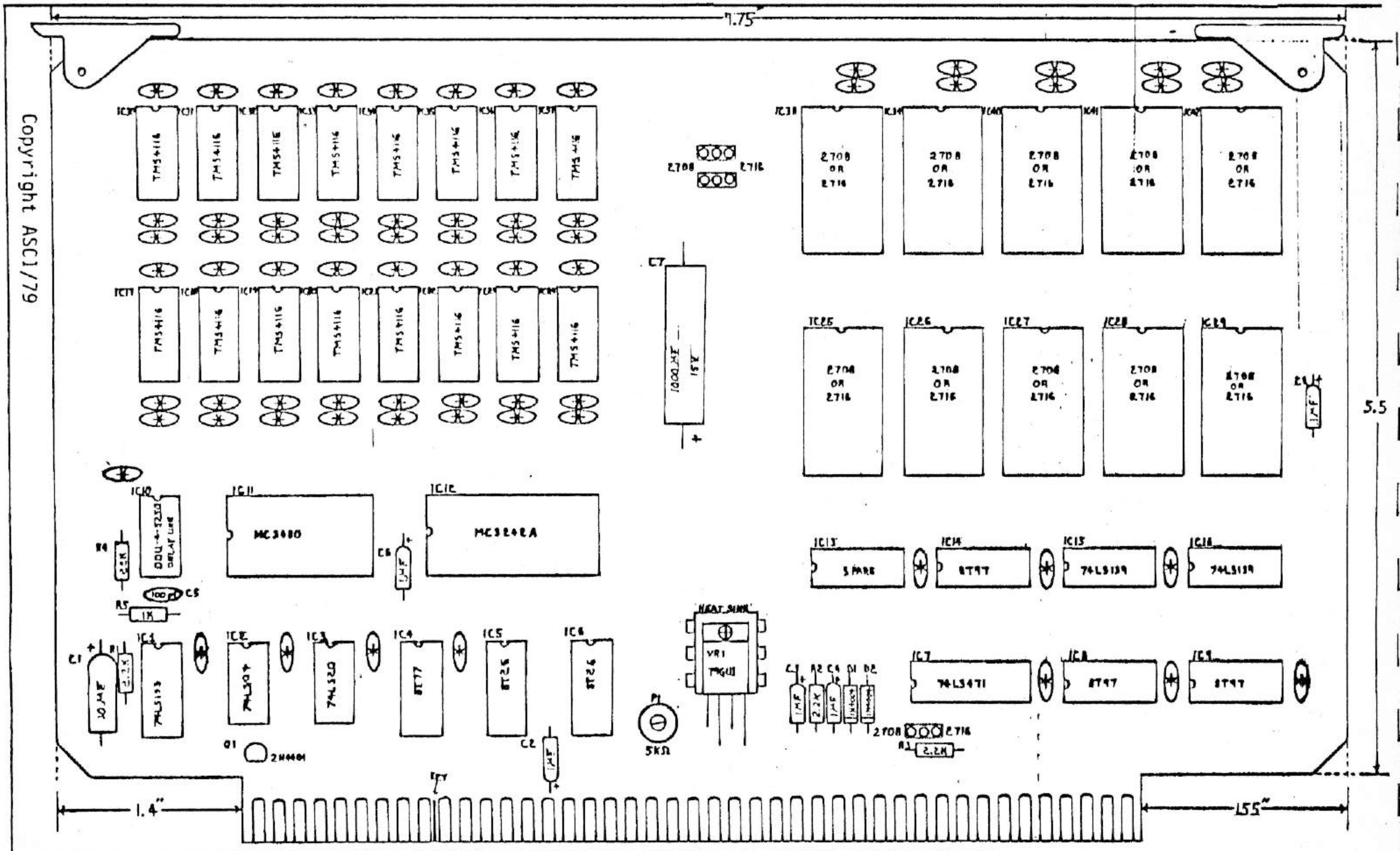
Copyright ASCI/79



Copyright ASCI/79



Copyright ASCI/79



REF.	DESCRIPTION	QUAN	REF	DESCRIPTION	QUAN	REF.	DESCRIPTION	QUAN
IC1	74LS193 COUNTER	1	IC25-29	2708 or 2716 PROM, INTEL OR T.I.		VR1	REGULATOR, FAIRCHILD µA78GU1CF	1
IC2	74LS04 INVERTER	1	IC38-42	2708 or 2716 PROM, " "	10	JUMPER PIN	AMP 85331-P	1
IC3	74LS20 GATE	1	R1, 2, 3	2.2K ¼W 5% RESISTOR	3	JUMPER	BERG 65474-001	3
IC4,8,9,14	8T97N 3 STATE BUFFER, SIGNETICS	4	R4	22K ¼W 5% "	1		HEAT SINK AHAM-TOR 364	1
IC5,6	8T26 BI-DIRECTIONAL BUFFER, "	2	R5	1KΩ ¼W 5% "	1		INSULATOR, PHENOLIC .75"x.75"x1/32	1
IC7	74S471 FUSE LINK ROM T.I.	1	Q1	2N4401 NPN TRANSISTOR	1		CARD EXTRACTOR, SCANBE S-203	2
IC10	DDU-4-5250 DATA DELAY DEVICES	1	C1	10µF @20V CAP, KEMET T110B106K020AS	1	PCB	ASCI LS/MB REV1	1
IC11	MC3480P CONTROLLER MOTOROLA	1	C2,3,4,6,8	1µF @35V CAP, KEMET T110A105K035AS	5	IC SOCKET	14 PIN DIP, AMP 640357-3	2
IC12	MC3242A MULTIPLEXER "	1	C5	100pF @1KV CAP, CENTRALAB DD-101	1	"	16 " " " 640358-3	25
IC13	SPARE	1	C7	1000µF @15V CAP, MALLORY TC1501B	1	"	20 " " " 640464-3	1
IC15,16	74LS139 2 TO 4 DECODER	2	BYPASS	.1µF @50V CAP, MEPCO C123-Z9J103M050	72	"	24 " " " 640361-3	1
IC17-24	TMS4116N DYNAMIC RAM T.I.		P1	5KΩ TRIMPOT, BOURNS 3339P-1-503	1	"	28 " " " 640362-3	1
IC30-37	TMS4116N	16	D1, D2	1N4004 DIODE	2	"	24 " " AUGAT 524-AG10D	10

ASCI/µ68

DATE 7/20/79

DRAWN BY LFM

ASCI DYNAMIC RAM LS/MB

LAYOUT

G18

```
00001                          NAM    T68BUG
00002                          TTL    RESIDENT MONITOR FOR ASCI TRAINER
0003                    *
0004                    * VERSION 4.1
00005                   *
00006                   * 19 MAR 81                     11:33 AM DX
00007                   *
00008                   * THIS VERSION USES THE LS/DR DIGITAL CASSETT.
00009                   *
00010                   *
00011                   *  T68BUG AND USER STACK AREAS
00012                   *
00013A 0000                    ORG    $0000
00014A 0000     000B   A       RMB    11        USER STACK AREA
00015A 000B     0006   A USTACK RMB   6
00016A 0011     0002   A USRPC  RMB   2         PROGRAM COUNTER, HIGH.
00017A 0013     0014   A BSTACK RMB   20        BUG STACK
00018           0026   A TOPSTK EQU   *-1
00019A 0027     0002   A USRSTK RMB   2         USER STACK POINTER
00020                   *
00021                   *  BRANCH TEST AREA FOR SINGLE STEP
00022                   *  AREA INITIALIZED BEFORE EACH TEST
00023                   *
00024A 0029     0001   A RAMOP  RMB   1         BXX
00025A 002A     0001   A RAMOFF RMB   1         *+5
00026A 002B     0001   A JMPFLS RMB   1         JMP
00027A 002C     0002   A FLSDES RMB   2         #FALSBR
00028A 002E     0001   A JMPTRU RMB   1         JMP
0029A 002F      0002   A TRUDES RMB   2         #TRUEBR
0030            002F   A SAVEX  EQU   TRUDES    TEMP STORE FOR X IN DISPLY
00031           002C   A TEMPX  EQU   FLSDES    TEMP STORE FOR X IN ADDABX
00032                   *
00033                   *  FLAGS,POINTERS,ECT.
00034                   *
00035A 0031     0002   A IRQADR RMB   2         IRQ ADR HOLDER
00036A 0033     0002   A SWIADR RMB   2         SWI ADR HOLDER
00037A 0035     0002   A NMIADR RMB   2         NMI ADR HOLDER
00038A 0037     0001   A MODFLG RMB   1         DO/EXAMINE/STEP FLAG
00039A 0038     0002   A ADDRSS RMB   2         ADDRESS HOLDER,
00040A 003A     0002   A BPADD  RMB   2         BREAK POINT ADDRESS
00041A 003C     0001   A BPDATA RMB   1         BREAK POINT DATA
00042A 003D     0001   A CHKSUM RMB   1         CHECK SUM STORE FOR LOAD/PUNCH
00043A 003E     0002   A BEGADD RMB   2         BEGINNING ADDRESS STORE FOR PUNCH
00044A 0040     0002   A ENDADD RMB   2         END ADDRESS STORE FOR PUNCH
00045A 0042     0001   A SBAUD  RMB   1         SAVE FOR BAUD WHILE LOAD/PUNCH
00046A 0043     0001   A BAUDCD RMB   1      FG BAUD RATE CODE
00047A 0044     0001   A NOWRF  RMB   1         NO-WRITE FLAG
00048A 0045     0001   A FOUND  RMB   1         PUSH BUTTON FOUND FLAG
00049A 0046     0001   A CHNGFL RMB   1         CHANGE FLAG
00050A 0047     0001   A INCHAR RMB   1         CHARACTER INPUT
00051A 0048     0002   A DIGPTR RMB   2         DIGIT POINTER
00052A 004A     0001   A DIGIT1 RMB   1         DIGIT CELL #1
00053A 004B     0001   A DIGIT2 RMB   1         DIGIT CELL #2
0054A 004C      0001   A DIGIT3 RMB   1         DIGIT CELL #3
0055A 004D      0001   A DIGIT4 RMB   1         DIGIT CELL #4
00056A 004E     0001   A DIGIT5 RMB   1         DIGIT CELL #5
00057A 004F     0001   A DIGIT6 RMB   1         DIGIT CELL #6
```

```
00059                      *
00060                      * TAPE ROUTINE RAM STORAGE
00061                      *
  062A 0050     0002  A DATPTR RMB     2          DATA POINTER LOAD/PUNCH
00063A 0052     0002  A COUNTR RMB     2          BYTES TO LOAD/PUNCH
00064A 0054     0008  A TBUFF  RMB     8          TAPE BUFFER
00065                      *
00066                      * TAPE EQUATES
00067                      *
00068           0000  A REWCMD EQU     0          REWIND TO START HOLE
00069           0001  A REDCMD EQU     1          READ 1 RECORD COMMNAD
00070           0002  A WRTCMD EQU     2          WRITE 1 RECORD COMMAND
00071           0003  A STACMD EQU     3          REQUEST STATUS COMMAND
00072           0006  A CLRCMD EQU     6          REWIND TO CLEAR LEADER
00073           5854  A XTAPE  EQU     $5854      DATA FOR EXTERNAL TAPE CHECK
00074           F000  A EXTTAP EQU     $F000      FROM ADDR FOR EXTERNAL CHECK
00075           F002  A XPUNCH EQU     EXTTAP+2 ADDR OF JUMP TO EXT PUNCH
00076           F005  A XLOAD  EQU     XPUNCH+3 ADDR OF JUMP TO EXT LOAD
00077           0008  A B9600  EQU     8          BAUD CONSTANT FOR 9600 BAUD
00078                      *
00079                      * IOFLAG DESCRIPTION
00080                      *
00081                      *     BIT           MEANING
00082                      *      7            CRT ACTIVE
00083                      *     6-2           NOT ASSIGNED
00084                      *      1            CARRIAGE RETURN
00085                      *      0            SPACE
00086                      *
  087A 005C     0001  A IOFLAG RMB     1          CRT/KEYPAD FLAG :00 => KEYPAD
   088                     *
00089                      *  I/O MEMORY ASSIGNMENTS
00090                      *
00091           E000  A KBOT07 EQU     $E000      KB DATA KEYS 0-7
00092           E001  A KBOCTL EQU     KBOT07+1 KB CONTROL KEYS 0-7
00093           E002  A KB8TOF EQU     KBOCTL+1 KB DATA KEYS 8-F
00094           E003  A KB8CTL EQU     KB8TOF+1 KB CONTROL KEYS 8-F

00096           E400  A LEDDIG EQU     $E400      LED DATA DIGITS
00097           E401  A DIGCTL EQU     LEDDIG+1 LED CONTROL DIGITS
00098           E402  A LEDSEG EQU     DIGCTL+1 LED DATA SEGMENTS
00099           E403  A SEGCTL EQU     LEDSEG+1 LED CONTROL SEGMENTS

00101           E800  A PIA3A  EQU     $E800      SPARE PIAA DATA
00102           E801  A PIA3AC EQU     PIA3A+1    SPARE PIAA CONTROL
00103           E802  A PIA3B  EQU     PIA3AC+1   SPARE PIAB DATA
00104           E803  A PIA3BC EQU     PIA3B+1    SPARE PIAB CONTROL

00106           EC00  A ACIACS EQU     $EC00      ACIA CONTROL/STATUS
00107           EC01  A ACIAXR EQU     ACIACS+1 ACIA TRANSMIT/RECEIVE
```

MSB 80 = CRT

```
0109                         *
0110                         * JUMP VECTORS TO USER ACCESSIBLE ROUTINES
0'                           *
011.  F800                          ORG     $F800
0113                         *
0114A F800 7E F874  A               JMP     RSTART    T68BUG RESTART
0115A F803 7E FD96  A               JMP     KEYIN     GET KEY FROM HEX KEYBOARD -> A
0116A F806 7E FD7C  A               JMP     IN2HEX    GET TWO HEX DIGITS FROM KEYBOARD -> A
0117A F809 7E FD47  A               JMP     GETADD    GET FOUR DIGITS FROM KEYBOARD -> ADDRSS
0118A F80C 7E FDB8  A               JMP     DISPLY    REFRESH DISPLAY FROM DIGIT1 - DIGIT6
0119A F80F 7E FE66  A               JMP     WRDATA    CONVERT HEX DIGIT TO DISPLAY CODE
0120A F812 7E FD5F  A               JMP     ADROUT    DISPLAY 4 DIGIT ADDRESS IN ADDRSS
0121A F815 7E FDDC  A               JMP     ADDABX    ADD (A,B) TO X
0122A F818 7E FDE9  A               JMP     ADDBX     ADD B TO X
0123A F81B 7E FC44  A               JMP     INEE      GET ONE CHARACTER FROM ACIA -> A
0124A F81E 7E FC31  A               JMP     INHEX     GET ONE HEX DIGIT FROM ACIA -> A
0125A F821 7E FC20  A               JMP     IN2       GET 2 HEX DIGITS FROM ACIA -> m(x)
~~0126A F824 7E FC1E  A               JMP     IN4       GET 4 HEX DIGITS FROM ACIA ->M(X)~~
0127A F827 7E FC6C  A               JMP     OUTEE     SEND BYTE IN A -> ACIA
0128A F82A 7E FC64  A               JMP     OUTHEX    SEND HEX DIGIT IN A -> ACIA
0129A F82D 7E FC53  A               JMP     OUT2      SEND 2 HEX DIGITS IN M(X) -> ACIA
~~0130A F830 7E FC51  A               JMP     OUT4      SEND 4 HEX DIGITS IN M(X) -> ACIA~~
0131A F833 7E FCB7  A               JMP     BAUD      INITIALIZE ACIA
0132A F836 7E FDF5  A               JMP     LPROM     PROM AUTO LOADER
0133A F839 7E FCD7  A               JMP     PDATAL    PRINT CHARACTER STRING
0134A F83C 7E FCD4  A               JMP     OUTCRL    CR/LF , 3 NULLS
0135A F83F 7E FA5A  A               JMP     LOADO     LOAD (COMPLETE)
01    F842 7E F9D5  A               JMP     PUNCHO    PUNCH (STALL,SACIA,REWIND,SAVE)
0     F845 7E F9DE  A               JMP     SAVE      TAPE SAVE 1 RECORD
01  H F848 7E FA29  A               JMP     PUNCHZ    GO2CLR,STALL,RACIA
0139A F84B 7E FE48  A (A) --->      JMP     ONDELA    12 SECOND STALL
0140A F84E 7E F8D3  A               JMP     BANNER    WARM START T68BUG
0141A F851 7E FA88  A               JMP     READ      TAPE LOAD 1 RECORD
0142A F854 7E FD24  A               JMP     SACIA     SET ACIA FOR TAPE
0143A F857 7E FD35  A               JMP     RACIA     RESTORE ACIA FOR T68BUG
0144A F85A 7E FABB  A               JMP     WAT4ST    WAIT FOR TAPE STATUS
0145A F85D 7E FD0C  A               JMP     OUTTAP    DATA TO TAPE, UPDATE CHKSUM
0146A F860 7E FCE3  A               JMP     INTAP     DATA FROM TAPE, UPDATE CHKSUM
0147A F863 7E FE89  A               JMP     PUT2      2 DIGITS TO DISP/CRT
0148A F866 7E FE9E  A               JMP     PUT4      4 DIGITS TO DISP/CRT
0149A F869 7E FE55  A               JMP     INPUT     GET KEY FROM KEYPAD/CRT
```

Ⓐ ACCESS FE4A w̄ "A" LOADED w̄ (# OF SEC X 10).

```
00151                     *
00152                     *  PROGRAM AREA BEGINS
00153                     *
  154A F86C DE 35      A NMI     LDX    NMIADR    GOTO NMI SERVICE ROUTINE
00155A F86E 6E 00      A         JMP    0,X       ADDRESS IS IN NMIADR
00156A F870 DE 31      A IRQ     LDX    IRQADR    GOTO IRQ SERVICE ROUTINE
00157A F872 6E 00      A         JMP    0,X       ADDRESS IS IN IRQADR
00158                     *
00159                     *  POWER UP/RESET VECTORED HERE
00160                     *
00161A F874 0F            RSTART SEI
00162A F875 8E 0026    A         LDS    #TOPSTK
00163A F878 CE 000B    A         LDX    #USTACK   SET DEFAULT USER STACK
00164A F87B DF 27      A         STX    USRSTK
00165A F87D CE FBE7    A         LDX    #NMISRV   NMI SERVICE
00166A F880 DF 35      A         STX    NMIADR    LOCATION
00167A F882 DF 31      A         STX    IRQADR    INITIALIZE IRQ LOCATION
00168A F884 CE FBEC    A         LDX    #PCCCEX   DEFAULT SWI SERVICE
00169A F887 DF 33      A         STX    SWIADR    INITIALIZE SWI LOCATION
00170A F889 4F                   CLRA             CLEAR
00171A F88A 97 44      A         STAA   NOWRF     NO WRITE FLAG AND
00172A F88C 97 45      A         STAA   FOUND     FOUND FLAG
00173A F88E 97 46      A         STAA   CHNGFL    RESET CHANGE FLAG
00174A F890 97 5C      A         STAA   IOFLAG
00175A F892 43                   COMA             SET BAUD RATE CODE FOR 110
00176A F893 97 43      A         STAA   BAUDCD
00177                     *
00178                     * INITIALIZE PIAS
00179                     *
   80A F895 4F                   CLRA             OPEN DATA DIRECTION REGISTERS
00181A F896 B7 E001    A         STAA   KBOCTL
00182A F899 B7 E003    A         STAA   KB8CTL
00183A F89C B7 E401    A         STAA   DIGCTL
00184A F89F B7 E403    A         STAA   SEGCTL
00185A F8A2 CE 0004    A         LDX    #$0004    SET KEYBOARD FOR INPUT
00186A F8A5 FF E000    A         STX    KBOTO7
00187A F8A8 FF E002    A         STX    KB8TOF
00188A F8AB CE FF04    A         LDX    #$FF04    SET DISPLAY FOR OUTPUT
00189A F8AE FF E400    A         STX    LEDDIG
00190A F8B1 FF E402    A         STX    LEDSEG
00191A F8B4 BD FCB7    A         JSR    BAUD
00192A F8B7 CE 0000    A INITBD  LDX    #0000     CLEAR BAUD RATE COUNTER
00193A F8BA B6 E401    A INITB1  LDAA   DIGCTL    IS CRT TALKING ?
00194A F8BD 49                   ROLA
00195A F8BE 25 10 F8D0           BCS    JBAUDR    YES, GO TO BAUD RATE ROUTINE
00196A F8C0 B6 E000    A         LDAA   KBOTO7    IS KEYPAD TALKING ?
00197A F8C3 88 FF      A         EORA   #$FF
00198A F8C5 26 0C F8D3           BNE    BANNER    YES, INITIALIZE DISPLAY
00199A F8C7 B6 E002    A         LDAA   KB8TOF    IS KEYPAD TALKING ?
00200A F8CA 88 FF      A         EORA   #$FF
00201A F8CC 27 EC F8BA           BEQ    INITB1    NO, KEEP WATCHING
00202A F8CE 20 03 F8D3           BRA    BANNER
00203A F8D0 BD FC79    A JBAUDR  JSR    BAUDRT
00204A F8D3 BD FDEF    A BANNER  JSR    SETDPT    SET DIGIT POINTER
0   05A F8D6 C6 04     A         LDAB   #4        WRITE
0  206A F8D8 BD FE9E   A         JSR    PUT4      'ASCI UP'
00207A F8DB BD FEE8    A         JSR    RQSCRL    IN THE
00208A F8DE C6 07      A         LDAB   #7        DIGIT
```

```
00209A F8E0 BD FE89   A          JSR   PUT2      CELLS
00210A F8E3 CE 0000   A          LDX   #0        DELAY
00211A F8E6 09          DLOOP    DEX
00212A F8E7 26 FD F8E6           BNE   DLOOP
00213A F8E9 7C 0044   A CONTRL   INC   NOWRF     INCREMENT NO WRITE FLAG
00214A F8EC 4F                   CLRA            RESET DO/EX/ST FLAG
00215A F8ED 97 37     A          STAA  MODFLG
00216A F8EF BD FE55   A CONTL1   JSR   INPUT     GET CONTROL CHARACTER
00217A F8F2 CE F8FE   A          LDX   #JMPTBL   GET JUMP TABLE BASE ADDRESS
00218A F8F5 16                   TAB
00219A F8F6 58                   ASLB            CALCULATE OFFSET
00220A F8F7 BD FDE9   A          JSR   ADDBX     GET TABLE ADDRESS
00221A F8FA EE 00     A          LDX   0,X       GET DESTINATION
00222A F8FC 6E 00     A          JMP   0,X       GO TO IT
00223                   *
00224                   *  CONTROL KEY JUMP TABLE
00225                   *
00226A F8FE      F953 A JMPTBL   FDB   AJMP,ABRA,SNGSTP,BPOINT
00227A F906      FBEC A          FDB   PCCCEX,XAEX,SPBEX,SETPC
00228A F90E      F9AA A          FDB   PUNCH,LOAD,STEP,BKWARD
00229A F916      F91E A          FDB   CHANGE,GO2USR,EXAMNE,FORWRD
```

```
00232                         *
00233                         *   CHANGE MEMORY
00234                         *
00235A F91E 96 46        A CHANGE LDAA    CHNGFL    TEST CHANGE FLAG
00236A F920 27 C7 F8E9            BEQ     CONTRL    THE CHANGE COMMAND
00237A F922 7F 0046      A        CLR     CHNGFL    RESET CHANGE FLAG
00238A F925 C6 00        A        LDAB    #0        WRITE OUT UNDERLINES IN
00239A F927 BD FE89      A        JSR     PUT2      DIGIT CELLS 5 & 6
00240A F92A CE 004E      A        LDX     #DIGIT5   POINT TO DIGIT 5
00241A F92D DF 48        A        STX     DIGPTR
00242A F92F BD FD7C      A        JSR     IN2HEX    GET 2 HEX DIGITS
00243A F932 DE 38        A        LDX     ADDRSS    GET ADDRESS
00244A F934 A7 00        A        STAA    0,X       PUT DATA IN MEMORY
00245A F936 A1 00        A        CMPA    0,X       IF THE WRITE
00246A F938 27 08 F942            BEQ     OK        WAS BAD
00247A F93A C6 03        A        LDAB    #3        WRITE OUT 2 HYPHENS
00248A F93C BD FE89      A        JSR     PUT2      TO THE DIGIT CELLS
00249A F93F 5F                    CLRB              CLEAR STEP
00250A F940 D7 37        A        STAB    MODFLG    MODE FLAG
00251A F942 96 37        A OK     LDAA    MODFLG    IF FLAG IS 'STEP' MODE
00252A F944 2E 03 F949            BGT     FORWRD    GOTO FORWARD
00253A F946 7E F8E9      A        JMP     CONTRL    GO BACK TO CONTROL
00254                         *
00255                         * MOVE THROUGH AND EXAMINE MEMORY COMMANDS
00256                         *
00257A F949 DE 38        A FORWRD LDX     ADDRSS    INCREMENT
00258A F94B 08                    INX               THE
00259A F94C 20 19 F967            BRA     NEXT

00261A F94E DE 38        A BKWARD LDX     ADDRSS    DECREMENT
00262A F950 09                    DEX               THE
00263A F951 20 14 F967            BRA     NEXT

00265A F953 DE 38        A AJMP   LDX     ADDRSS    GET CURRENT ADDRESS
00266A F955 09                    DEX               BACK UP ONE BYTE
00267A F956 EE 00        A        LDX     0,X       GET THE CONTENTS
00268A F958 20 0D F967            BRA     NEXT

00270A F95A DE 38        A ABRA   LDX     ADDRSS    GET CURRENT ADDRESS
00271A F95C E6 00        A        LDAB    0,X       GET CONTENTS
00272A F95E 08                    INX               POINT TO NEXT BYTE
00273A F95F 4F                    CLRA              SET UP OFFSET
00274A F960 5D                    TSTB              NEGATIVE?
00275A F961 2A 01 F964            BPL     ABRA1     NO
00276A F963 43                    COMA              YES
00277A F964 BD FDDC      A ABRA1  JSR     ADDABX    CALCULATE THE OFFSET
00278A F967 DF 38        A NEXT   STX     ADDRSS    ADDRESS
00279A F969 BD FEF2      A        JSR     DOCRLF    DO A CR/LF
00280A F96C BD FD5F      A        JSR     ADROUT    DISPLAY ADDRESS
00281A F96F 20 16 F987            BRA     CONTNS    GO OUTPUT CONTENTS

00283A F971 7C 0037      A STEP   INC     MODFLG    SET FLAG TO STEP
00284A F974 20 06 F97C            BRA     EXAMNE    GO SHARE CODE

00286A F976 7A 0037      A SETPC  DEC     MODFLG    SET FLAG TO "SET PC"
00287A F979 C6 08        A        LDAB    #8        DISPLAY "PC"
00288A F97B    8C        A        FCB     $8C       SKIP 2
```

```
00290A F97C C6 09      A EXAMNE LDAB   #9         DISPLAY " "
00291A F97E BD FEF2    A        JSR    DOCRLF     DO CR/LF
00292A F981 BD FE89    A        JSR    PUT2
00293A F984 BD FD47    A        JSR    GETADD
00294A F987 7C 0046    A CONTNS INC    CHNGFL     SET CHANGE FLAG
00295A F98A DE 38      A        LDX    ADDRSS     GET FULL 16 BIT ADDRESS
00296A F98C 96 37      A        LDAA   MODFLG     IF THE FLAG IS NOT 'DO'
00297A F98E 2A 0D F99D          BPL    NOTDO      SKIP TO NOTDO
00298A F990 DE 27      A        LDX    USRSTK     GET USER STACK POINTER
00299A F992 96 38      A        LDAA   ADDRSS
00300A F994 A7 06      A        STAA   6,X        SET HIGH BYTE OF PC
00301A F996 96 39      A        LDAA   ADDRSS+1
00302A F998 A7 07      A        STAA   7,X        SET LOW BYTE OF PC
00303A F99A 7E F8E9    A        JMP    CONTRL
00304A F99D 27 03 F9A2 NOTDO    BEQ    NOTDO1     IF NOT STEP MODE, CONTINUE
00305A F99F 7E F91E    A        JMP    CHANGE     ELSE GOTO CHANGE
00306A F9A2 A6 00      A NOTDO1 LDAA   0,X        FETCH DATA
00307A F9A4 BD FD6B    A        JSR    OUT2HX     OUTPUT 2 HEX DIGITS
00308A F9A7 7E F8E9    A        JMP    CONTRL
```

```
00310                        *
00311                        *  TAPE HANDLING ROUTINES
00312                        *
00313                        *  TAPE PUNCH ROUTINE
00314                        *
00315A F9AA FE F000    A PUNCH  LDX    EXTTAP    EXTERNAL TAPE ROUTINE?
00316A F9AD 8C 5854    A        CPX    #XTAPE
00317A F9B0 26 03 F9B5          BNE    *+5       NO, SKIP NEXT INSTRUCTION
00318A F9B2 7E F002    A        JMP    XPUNCH    JUMP TO EXTERNAL ROUTINE
00319A F9B5 BD FEF2    A        JSR    DOCRLF    DO CR/LF
00320A F9B8 C6 05      A        LDAB   #5        "BEGIN"
00321A F9BA BD FD3C    A        JSR    RQSADD    INPUT BEGINNING ADDRESS
00322A F9BD DF 3E      A        STX    BEGADD
00323A F9BF C6 06      A        LDAB   #6        "END"
00324A F9C1 BD FD3C    A        JSR    RQSADD    INPUT END ADDRESS
00325A F9C4 DF 40      A        STX    ENDADD
00326A F9C6 C6 01      A        LDAB   #1        BLANK DIGITS 5 & 6
00327A F9C8 BD FE89    A        JSR    PUT2
00328A F9CB 8D 08 F9D5          BSR    PUNCHO    DO THE SAVE
00329A F9CD 8D 5A FA29          BSR    PUNCHZ    TAPE OFF, DELAY, RESET ACIA
00330A F9CF BD FA7E    A        JSR    TPDONE    DONE MESSAGE
00331A F9D2 7E F8E9    A        JMP    CONTRL
00332                        *
00333A F9D5 BD FE48    A PUNCHO JSR    ONDELA    WAIT FOR SWITCH
00334A F9D8 BD FD24    A        JSR    SACIA     SET ACIA
00335A F9DB BD FAB3    A        JSR    REWIND    REWIND THE TAPE
00336                        *
00337                        * SAVE 1 BLOCK OF DATA TO THE TAPE
00338                        *
00339A F9DE 86 02      A SAVE   LDAA   #WRTCMD
00340A F9E0 97 54      A        STAA   TBUFF     WRITE COMMAND INTO BUFFER
00341A F9E2 DE 40      A        LDX    ENDADD    CALCULATE
00342A F9E4 96 3E      A        LDAA   BEGADD    THE
00343A F9E6 D6 3F      A        LDAB   BEGADD+1  SIZE
00344A F9E8 43                  COMA             OF
00345A F9E9 53                  COMB             THE
00346A F9EA CB 01      A        ADDB   #1        BLOCK
00347A F9EC 89 00      A        ADCA   #0
00348A F9EE BD FDDC    A        JSR    ADDABX
00349A F9F1 08                  INX              ADD 1 END TEST
00350A F9F2 DF 59      A        STX    TBUFF+5   SIZE INTO BUFFER
00351A F9F4 C6 05      A        LDAB   #5        ADD OVERHEAD
00352A F9F6 BD FDE9    A        JSR    ADDBX     FOR
00353A F9F9 DF 55      A        STX    TBUFF+1   BYTES TO SAVE.
00354A F9FB DE 3E      A        LDX    BEGADD    START ADDRESS INTO BUFFER
00355A F9FD DF 57      A        STX    TBUFF+3
00356A F9FF CE 0003    A        LDX    #3        SEND
00357A FA02 DF 52      A        STX    COUNTR    THE
00358A FA04 CE 0054    A        LDX    #TBUFF    WRITE COMMAND
00359A FA07 BD FD0C    A        JSR    OUTTAP    AND SIZE.
00360A FA0A 7F 003D    A        CLR    CHKSUM    CHKSUM = 0
00361A FA0D CE 0004    A        LDX    #4        WRITE A HEADER
00362A FA10 DF 52      A        STX    COUNTR    OF THE
00363A FA12 CE 0057    A        LDX    #TBUFF+3  START ADDRESS
00364A FA15 BD FD0C    A        JSR    OUTTAP    AND THE SIZE+1.
0365A FA18 DE 59       A        LDX    TBUFF+5   THEN
00366A FA1A DF 52      A        STX    COUNTR    SAVE
00367A FA1C 27 05 FA23          BEQ    SAVE1
```

```
00368A FA1E DE 3E     A          LDX     BEGADD    THE
00369A FA20 BD FD0C   A          JSR     OUTTAP    DATA.
00370A FA23 96 3D     A SAVE1    LDAA    CHKSUM    SAVE
00371A FA25 40                   NEGA              THE NEGATIVE
00372A FA26 7E FC6C   A          JMP     OUTEE     CHECKSUM, AND RETURN
00373                   *
00374                   * REWIND TO CLEAR LEADER, WAIT FOR SWITCH
00375                   * RESET THE BAUD RATE AND B = "DONE" MESSAGE.
00376                   *
00377A FA29 BD FAB6   A PUNCHZ JSR     GO2CLR    REWIND TO CLEAR LEADER
00378A FA2C BD FE48   A          JSR     ONDELA    WAIT FOR SWITCH
00379A FA2F BD FD35   A          JSR     RACIA     RESTORE ACIA
00380A FA32 C6 00     A          LDAB    #0        "DONE" MESSAGE
00381A FA34 39                   RTS
```

```
00383                          *
00384                          *   TAPE LOAD ROUTINE
0385                           *
0386A FA35 FE F000   A LOAD     LDX     EXTTAP    IS THERE AN EXTERNAL
00387A FA38 8C 5854  A          CPX     #XTAPE    TAPE ROUTINE ?
00388A FA3B 26 03 FA40          BNE     *+5       NO, SKIP NEXT INSTRUCTION
00389A FA3D 7E F005  A          JMP     XLOAD
00390A FA40 BD FEF2  A          JSR     DOCRLF    DO CR/LF
00391A FA43 C6 01    A          LDAB    #1        DISPLAY "LOAD"
00392A FA45 BD FE9E  A          JSR     PUT4
00393A FA48 C6 01    A          LDAB    #1        BLANK DIGITS 5 & 7
00394A FA4A BD FE89  A          JSR     PUT2
00395A FA4D 86 64    A          LDAA    #100      STALL TO LET HIM
00396A FA4F BD FDB8  A LWAIT    JSR     DISPLY    SEE THE LOAD MESSAGE
00397A FA52 4A                  DECA
00398A FA53 26 FA FA4F          BNE     LWAIT
00399A FA55 8D 03 FA5A          BSR     LOAD0
00400A FA57 7E F8E9  A          JMP     CONTRL
00401                          *
00402A FA5A BD FE48  A LOAD0    JSR     ONDELA    WAIT FOR SWITCH
00403A FA5D BD FD24  A          JSR     SACIA     SETUP ACIA FOR TAPE
00404A FA60 8D 51 FAB3          BSR     REWIND    REWIND TO HOLE
00405A FA62 8D 24 FA88          BSR     READ      READ TAPE
00406A FA64 24 06 FA6C          BCC     LOAD1
00407A FA66 86 01    A          LDAA    #1        FORCE CHECKSUM WRONG
00408A FA68 97 3D    A          STAA    CHKSUM
00409A FA6A 20 02 FA6E          BRA     LOAD2
00410A FA6C 8D 48 FAB6 LOAD1    BSR     GO2CLR    REWIND TO CLEAR LEADER
0411A FA6E BD FE48   A LOAD2    JSR     ONDELA    WAIT FOR SWITCH
00412A FA71 BD FD35  A          JSR     RACIA     RESTORE ACIA
00413A FA74 7D 003D  A          TST     CHKSUM    TEST CHECKSUM AND
00414A FA77 27 03 FA7C          BEQ     LOAD3     PRINT APPROPRIATE MESSAGE.
00415A FA79 C6 02    A          LDAB    #2        "BAD" LOAD
00416A FA7B    8C    A          FCB     $8C       SKIP 2 BYTES
00417A FA7C C6 03    A LOAD3    LDAB    #3        "GOOD" LOAD
00418A FA7E 96 5C    A TPDONE LDAA    IOFLAG
00419A FA80 8A 02    A          ORAA    #2        REQUEST CR/LF
00420A FA82 97 5C    A          STAA    IOFLAG
00421A FA84 BD FE9E  A          JSR     PUT4
00422A FA87 39                  RTS
00423                          *
00424                          * READ ONE RECORD FROM TAPE
00425                          *
00426                          * ENTRY: BEGADD MUST BE VALID
00427                          *        ENDADD MUST BE VALID
00428                          * EXIT:  CARRY = 1 ABORTED
00429                          *        CARRY = 0 NOT ABORTED
00430                          *        CHKSUM VALID
00431                          *
00432A FA88 86 01    A READ     LDAA    #REDCMD   SEND THE
00433A FA8A BD FC6C  A          JSR     OUTEE     READ COMMAND.
00434A FA8D 7F 003D  A          CLR     CHKSUM    CHECKSUM = 0
00435A FA90 CE 0004  A          LDX     #4        READ FIRST
00436A FA93 DF 52    A          STX     COUNTR    FOUR
437A FA95 CE 0054    A          LDX     #TBUFF    CHARACTERS
00438A FA98 BD FCE3  A          JSR     INTAP     INTO BUFFER.
00439A FA9B 25 15 FAB2          BCS     READ1     IF CARRY SET INVALID
00440A FA9D DE 56    A          LDX     TBUFF+2   GET SIZE
```

```
00441A FA9F 27 09 FAAA          BEQ    READ0     NO DATA READ CHKSUM
00442A FAA1 DF 52       A       STX    COUNTR    OF DATA BLOCK,
00443A FAA3 DE 54       A       LDX    TBUFF     POINT TO DESTINATION,
00444A FAA5 BD FCE3     A       JSR    INTAP     AND READ THE DATA IN.
00445A FAA8 25 08 FAB2          BCS    READ1     ABORT ?
00446A FAAA BD FC44     A READ0 JSR    INEE      ADD IN
00447A FAAD 9B 3D       A       ADDA   CHKSUM    THE
00448A FAAF 97 3D       A       STAA   CHKSUM    CHECK SUM.
00449A FAB1 0C                  CLC
00450A FAB2 39            READ1 RTS
```

```
00452                          *
00453                          * REWIND THE TAPE
00454                          *
00455A FAB3 86 00      A REWIND LDAA    #REWCMD  REWIND, GO TO START HOLE
00456A FAB5    8C      A        FCB     $8C      SKIP 2 BYTES
00457A FAB6 86 06      A GO2CLR LDAA    #CLRCMD  GOT TO CLEAR LEADER.
00458A FAB8 BD FC6C    A        JSR     OUTEE
00459A FABB C6 03      A WAT4ST LDAB    #3       NOW, WAIT
00460A FABD CE 0054    A        LDX     #TBUFF   FOR THE THREE
00461A FAC0 86 03      A        LDAA    #STACMD  REQUEST STATUS
00462A FAC2 BD FC6C    A        JSR     OUTEE
00463A FAC5 BD FC44    A REWIN1 JSR     INEE     STATUS
00464A FAC8 A7 00      A        STAA    0,X      CHARACTERS
00465A FACA 08                  INX              TO COME
00466A FACB 5A                  DECB             BACK.
00467A FACC 26 F7 FAC5          BNE     REWIN1
00468A FACE 39                  RTS
```

```
00470                        *
00471                        *  SINGLE STEP ROUTINE-OP CODE IS FIRST
00472                        *
 0473                        *
00474A FACF DE 27      A SNGSTP LDX     USRSTK   GET USER STACK POINTER
00475A FAD1 EE 06      A        LDX     6,X      GET RTI PC
00476A FAD3 A6 00      A        LDAA    0,X      GET OP CODE
00477A FAD5 97 3C      A        STAA    BPDATA   SAVE IT
00478A FAD7 CE FF16    A        LDX     #VALOP   ADDRESS OF TABLE
00479A FADA A1 00      A VALCHK CMPA    0,X
00480A FADC 27 3E FB1C          BEQ     NOTVLD   MATCH-INVALID CODE
00481A FADE 08                  INX              LOOK AT NEXT
00482A FADF 8C FF51    A        CPX     #EVALOP  TABLE EXHAUSTED?
00483A FAE2 26 F6 FADA          BNE     VALCHK   NO-LOOK AGAIN
00484A FAE4 81 20      A        CMPA    #$20     DE-CODE NUMBER OF BYTES
00485A FAE6 25 31 FB19          BCS     BYTES1   ONE BYTE CODE
00486A FAE8 81 30      A        CMPA    #$30
00487A FAEA 24 03 FAEF          BCC     *+5
00488A FAEC 7E FB7E    A        JMP     BRANCH   BRANCH INSTRUCTIONS
00489A FAEF 81 60      A        CMPA    #$60
00490A FAF1 25 26 FB19          BCS     BYTES1   ONE BYTE CODE
00491A FAF3 81 70      A        CMPA    #$70
00492A FAF5 25 1F FB16          BCS     BYTES2   TWO BYTE CODE
00493A FAF7 81 80      A        CMPA    #$80
00494A FAF9 25 18 FB13          BCS     BYTES3   THREE BYTE CODE
00495A FAFB 81 8C      A        CMPA    #$8C
00496A FAFD 27 14 FB13          BEQ     BYTES3   SPECIAL THREE BYTE
00497A FAFF 81 8E      A        CMPA    #$8E
00498A FB01 27 10 FB13          BEQ     BYTES3   SPECIAL THREE BYTE
 0499A FB03 81 B0      A        CMPA    #$B0
00500A FB05 25 0F FB16          BCS     BYTES2   TWO BYTE CODE
00501A FB07 81 C0      A        CMPA    #$C0
00502A FB09 25 08 FB13          BCS     BYTES3   THREE BYTE CODE
00503A FB0B 81 CE      A        CMPA    #$CE
00504A FB0D 27 04 FB13          BEQ     BYTES3   SPECIAL THREE BYTE
00505A FB0F 81 F0      A        CMPA    #$F0
00506A FB11 25 03 FB16          BCS     BYTES2   TWO BYTE CODE
00507A FB13 7E FB6A    A BYTES3 JMP     THREE    HANDLE THREE BYTE CODE
00508A FB16 7E FB4B    A BYTES2 JMP     TWO      HANDLE TWO BYTE CODE
00509A FB19 7E FB20    A BYTES1 JMP     ONE      HANDLE ONE BYTE CODES
00510A FB1C C6 02      A NOTVLD LDAB    #2       DISPLAY"??"
00511A FB1E BD FE89    A        JSR     PUT2
00512A FB21 DE 27      A        LDX     USRSTK   DISPLAY ADDRESS
00513A FB23 EE 06      A        LDX     6,X
00514A FB25 DF 38      A        STX     ADDRSS   SAVE ADDRESS
00515A FB27 BD FD5F    A        JSR     ADROUT
00516A FB2A 7E F8E9    A        JMP     CONTRL   RETURN TO CONTROL
00517                        *
00518                        *  ONE BYTE INSTRUCTIONS - OP CODE IN A
00519                        *
00520A FB2D DE 27      A ONE    LDX     USRSTK   GET USERS STACK IN X
00521A FB2F 81 39      A        CMPA    #$39     RTS?
00522A FB31 27 11 FB44          BEQ     RTSONE   YES
00523A FB33 81 3B      A        CMPA    #$3B     RTI?
  524A FB35 27 10 FB47          BEQ     RTIONE   YES
00525A FB37 81 3F      A        CMPA    #$3F     SWI?
00526A FB39 26 05 FB40          BNE     ONE1     NONE OF THE ABOVE
00527A FB3B DE 33      A        LDX     SWIADR   GET USER SWI
```

```
00528A FB3D 7E FBB8  A         JMP    BRKPNT    SET BREAKPOINT AND EXECUTE
00529A FB40 EE 06    A ONE1    LDX    6,X       SERVICE IT
00530A FB42 08                 INX
00531A FB43    8C    A         FCB    $8C       SKIP TWO BYTES
00532A FB44 EE 08    A RTSONE  LDX    8,X       GET RETURN ADDRESS
00533A FB46    8C    A         FCB    $8C       SKIP TWO BYTES
00534A FB47 EE 0D    A RTIONE  LDX    13,X      GET RTI ADDRESS
00535A FB49 20 6D FBB8         BRA    BRKPNT    SET BREAKPOINT AND EXECUTE
00536                  *
00537                  *  TWO BYTE INSTRUCTIONS - OP CODE IN A
00538                  *
00539A FB4B DE 27    A TWO     LDX    USRSTK    GET USERS STACK IN X
00540A FB4D EE 06    A         LDX    6,X       GET PC
00541A FB4F E6 01    A         LDAB   1,X       OPERAND IN B
00542A FB51 81 6E    A         CMPA   #$6E      JMP INDEXED?
00543A FB53 27 0C FB61         BEQ    INDJMP    YES
00544A FB55 81 AD    A         CMPA   #$AD      JSR INDEXED?
00545A FB57 27 08 FB61         BEQ    INDJMP    YES
00546A FB59 81 8D    A         CMPA   #$8D      BSR?
00547A FB5B 27 41 FB9E         BEQ    TRUEBR    YES
00548A FB5D 08                 INX              ADD TWO TO PROGRAM COUNTER
00549A FB5E 08                 INX              COUNTER IN X
00550A FB5F 20 57 FBB8         BRA    BRKPNT    SET BREAKPOINT AND EXECUTE
00551A FB61 DE 27    A INDJMP  LDX    USRSTK    B CONTAINS OFFSET
00552A FB63 EE 04    A         LDX    4,X       GET USER INDEX REGISTER
00553A FB65 BD FDE9  A         JSR    ADDBX     X+B->X
00554A FB68 20 4E FBB8         BRA    BRKPNT    SET BREAKPOINT AND EXECUTE
00555                  *
00556                  * THREE BYTE INSTRUCTIONS - OP CODE IN A
00557                  *
00558A FB6A DE 27    A THREE   LDX    USRSTK    GET USER STACK IN X
00559A FB6C EE 06    A         LDX    6,X       GET USER PC
00560A FB6E 81 7E    A         CMPA   #$7E      JMP?
00561A FB70 27 08 FB7A         BEQ    EXTJMP    YES
00562A FB72 81 BD    A         CMPA   #$BD      JSR?
00563A FB74 27 04 FB7A         BEQ    EXTJMP    YES
00564A FB76 08                 INX              ADD THREE TO
00565A FB77 08                 INX              PROGRAM COUNTER
00566A FB78 08                 INX              IN X
00567A FB79    8C    A         FCB    $8C       SKIP NEXT TWO BYTES
00568A FB7A EE 01    A EXTJMP  LDX    1,X       GET JMP DESTINATION
00569A FB7C 20 3A FBB8         BRA    BRKPNT    SET BREAKPOINT AND EXECUTE
00570                  *
00571                  *  BRANCH INSTRUCTIONS - OP CODE IN A
00572                  *
00573                  *  EIGHT BYTES IN RAM ARE EXECUTED TO
00574                  *  DETERMINE DIRECTION OF BRANCH
00575                  *
00576A FB7E 97 29    A .BRANCH STAA   RAMOP     INSERT OP CODE
00577A FB80 86 7E    A         LDAA   #$7E      JMP INSTRUCTION
00578A FB82 97 2B    A         STAA   JMPFLS    BRANCH FALSE
00579A FB84 97 2E    A         STAA   JMPTRU    BRANCH TRUE
00580A FB86 86 03    A         LDAA   #3        BRANCH OFFSET
00581A FB88 97 2A    A         STAA   RAMOFF
 0582A FB8A CE FB9E  A         LDX    #TRUEBR   BRANCH GOES
 0583A FB8D DF 2F    A         STX    TRUDES
00584A FB8F CE FBA7  A         LDX    #FALSBR   BRANCH NOT
00585A FB92 DF 2C    A         STX    FLSDES
```

```
00586A FB94 DE 27      A           LDX    USRSTK    GET USER STACK IN X
00587A FB96 A6 01      A           LDAA   1,X       CONDITION CODE
00588A FB98 EE 06      A           LDX    6,X       GET USER PC IN X
00589A FB9A 06                     TAP
00590A FB9B 7E 0029    A           JMP    RAMOP     EXECUTE THE BRANCH
00591                        *
00592                        *  BRANCH OCCURS IF RETURN IS HERE
00593                        *
00594A FB9E 4F               TRUEBR CLRA             CLEAR MSB OF OFFSET
00595A FB9F E6 01      A           LDAB   1,X       GET APERAND IN B
00596A FBA1 2A 01 FBA4             BPL    *+3
00597A FBA3 43                     COMA             MAKE WHOLE OFFSET NEGATIVE
00598A FBA4 BD FDDC    A           JSR    ADDABX    ADD OFFSET TO PC
00599                        *
00600                        *  BRANCH DOES NOT OCCUR IF RETURN IS HERE
00601                        *
00602A FBA7 08               FALSBR INX              CORRECT PC FOR OP CODE
00603A FBA8 08                     INX              AND OPERAND
00604A FBA9 20 0D FBB8             BRA    BRKPNT
```

```
00606                      *
00607                      * BREAKPOINT COMMAND ROUTINE
00608                      *
 0609A FBAB C6 04      A BPOINT LDAB    #4         DISPLAY "BP"
00610A FBAD BD FEF2    A        JSR     DOCRLF
00611A FBB0 BD FE89    A        JSR     PUT2
00612A FBB3 BD FD47    A        JSR     GETADD     GET ADDRESS
00613A FBB6 DE 38      A        LDX     ADDRSS
00614A FBB8 DF 3A      A BRKPNT STX     BPADD      SAVE BP ADDRESS
00615A FBBA A6 00      A        LDAA    X          GET CURRENT CONTENTS
00616A FBBC 97 3C      A        STAA    BPDATA     SAVE INSTRUCTION
00617A FBBE 86 3F      A        LDAA    #$3F       INSERT SWI
00618A FBC0 A7 00      A        STAA    X          IN CODE
00619A FBC2 9E 27      A GO2USR LDS     USRSTK     USING USERS STACK POINTER
00620A FBC4 3B                  RTI                GO EXECUTE HIS PROGRAM
```

```
00622                        *
00623                        *  SOFTWARE INTERRUPT SERVICE ROUTINE
00624                        *  IF ENTRY VIA BREAKPOINT OR SINGLE STEP
00625                        *  OP-CODE IS RESTORED
00626                        *
00627A FBC5 9F 27     A SWISRV STS    USRSTK   SAVE USER STACK POINTER
00628A FBC7 8E 0026   A        LDS    #TOPSTK  GET MONITOR STACK
00629A FBCA DE 27     A        LDX    USRSTK   USER SP ->X
00630A FBCC EE 06     A        LDX    6,X      USER PC->X
00631A FBCE 09                 DEX             ADJUST TO CHECK
00632A FBCF 9C 3A     A        CPX    BPADD    WAS IT A BREAK - POINT?
00633A FBD1 26 10 FBE3         BNE    SWI1     NO
00634A FBD3 96 3C     A        LDAA   BPDATA   RESTORE OP-CODE
00635A FBD5 A7 00     A        STAA   0,X
00636A FBD7 DE 27     A        LDX    USRSTK   DECREMENT THE PROGRAM COUNTER
00637A FBD9 6D 07     A        TST    7,X
00638A FBDB 26 02 FBDF         BNE    *+4
00639A FBDD 6A 06     A        DEC    6,X
00640A FBDF 6A 07     A        DEC    7,X
00641A FBE1 20 09 FBEC         BRA    PCCCEX
00642A FBE3 DE 33     A SWI1   LDX    SWIADR   GET USER SWI HANDLER VECTOR
00643A FBE5 6E 00     A        JMP    0,X      EXECUTE THE SWI HANDLER
00644                        *
00645                        *  NMI AND IRQ SERVICE ROUTINES
00646                        *
00647A FBE7 9F 27     A NMISRV STS    USRSTK   SAVE USER STACK POINTER
00648A FBE9 8E 0026   A        LDS    #TOPSTK
00649A FBEC DE 27     A PCCCEX LDX    USRSTK   USER SP->X
00650A FBEE A6 01     A        LDAA   1,X      GET CC IN A
00651A FBF0 EE 06     A        LDX    6,X      USER PC->X
00652A FBF2 C6 07     A        LDAB   #7       "PCCC"
00653A FBF4 20 10 FC06         BRA    DISPLA   DISPLAY PC AND CC
00654A FBF6 DE 27     A XAEX   LDX    USRSTK   USER SP->X
00655A FBF8 A6 03     A        LDAA   3,X      USER A->A
00656A FBFA EE 04     A        LDX    4,X      USER X->X
00657A FBFC C6 08     A        LDAB   #8       "X A "
00658A FBFE 20 06 FC06         BRA    DISPLA   DISPLAY X AND A
00659A FC00 DE 27     A SPBEX  LDX    USRSTK   USER SP->X
00660A FC02 A6 02     A        LDAA   2,X      USER B->A
00661A FC04 C6 09     A        LDAB   #9       "SP B"
00662A FC06 DF 38     A DISPLA STX    ADDRSS   SAVE FOR DISPLAY
00663A FC08 36                 PSHA
00664A FC09 37                 PSHB
00665A FC0A BD FEE8   A        JSR    RQSCRL   CR/LF
00666A FC0D BD FECD   A        JSR    CHKIOF
00667A FC10 33                 PULB
00668A FC11 BD FE9E   A        JSR    PUT4     PRINT THE MESSAGE
00669A FC14 BD FD5F   A        JSR    ADROUT   DISPLAY SELECTED REGISTER
00670A FC17 32                 PULA
00671A FC18 BD FD6B   A        JSR    OUT2HX
00672A FC1B 7E F8E9   A        JMP    CONTRL
```

```
00674                        *
00675                        * SERIAL I/O ROUTINES
 0676                        *
 0677                        *  INPUT HEX DIGITS
00678                        *
00679A FC1E 8D 00 FC20 IN4      BSR     IN2
00680A FC20 8D 0F FC31 IN2      BSR     INHEX     GET FIRST CHARACTER
00681A FC22 48                  ASLA
00682A FC23 48                  ASLA
00683A FC24 48                  ASLA
00684A FC25 48                  ASLA
00685A FC26 A7 00      A        STAA    0,X
00686A FC28 8D 07 FC31          BSR     INHEX     GET SECOND CHARACTER
00687A FC2A 84 0F      A        ANDA    #15       CONVERT TO HEX
00688A FC2C AB 00      A        ADDA    0,X       BUILD BYTE
00689A FC2E A7 00      A        STAA    0,X       SAVE IT
00690A FC30 39                  RTS
00691                        *
00692                        *  CONVERT INPUT CHARACTER FROM ASCII TO HEX
00693                        *
00694A FC31 8D 11 FC44 INHEX    BSR     INEE      GET BYTE
00695A FC33 84 7F      A        ANDA    #$7F
00696A FC35 80 30      A        SUBA    #$30      CONVERT FROM ASCII TO HEX
00697A FC37 25 F8 FC31          BCS     INHEX     NOT A HEX CHARACTER
00698A FC39 81 0A      A        CMPA    #$A
00699A FC3B 25 06 FC43          BCS     INHEX1
00700A FC3D 80 07      A        SUBA    #7
00701A FC3F 81 10      A        CMPA    #$10
 0702A FC41 24 EE FC31          BCC     INHEX     NOT $A - $F, SO NOT HEX
 0703A FC43 39         INHEX1   RTS
00704                        *
00705                        * INPUT BYTE
00706                        *
00707A FC44 37         INEE     PSHB              SAVE B
00708A FC45 C6 01      A        LDAB    #1
00709A FC47 F5 EC00    A INEE1  BITB    ACIACS    WAIT FOR RBF
00710A FC4A 27 FB FC47          BEQ     INEE1
00711A FC4C B6 EC01    A        LDAA    ACIAXR    GET THE CHARACTER
00712A FC4F 33                  PULB              RESTORE B
00713A FC50 39                  RTS
00714                        *
00715                        *  OUTPUT HEX DIGITS
00716                        *
00717A FC51 8D 00 FC53 OUT4     BSR     OUT2
00718A FC53 A6 00      A OUT2   LDAA    0,X       GET BYTE
00719A FC55 36                  PSHA              SAVE FOR CHECKSUM
00720A FC56 36                  PSHA              SAVE FOR SECOND DIGIT
00721A FC57 44                  LSRA
00722A FC58 44                  LSRA
00723A FC59 44                  LSRA
00724A FC5A 44                  LSRA
00725A FC5B 8D 07 FC64          BSR     OUTHEX
00726A FC5D 32                  PULA              RESTORE BYTE IN A
00727A FC5E 84 0F      A        ANDA    #15
  728A FC60 8D 02 FC64          BSR     OUTHEX
00729A FC62 32                  PULA
00730A FC63 39                  RTS
00731                        *
```

```
00732                        *  CONVERT HEX TO ASCII AND SEND OUT
00733                        *
00734A FC64 8B 30      A OUTHEX ADDA    #$30      ADD FIRST CONVERSION
 0735A FC66 81 39      A        CMPA    #$39      <= 9
00736A FC68 23 02 FC6C          BLS     OUTEE     YES
00737A FC6A 8B 07      A        ADDA    #7        NO - CONVERT A - F
00738A FC6C 37         OUTEE    PSHB              SAVE B
00739A FC6D C6 02      A        LDAB    #2
00740A FC6F F5 EC00    A OUTEE1 BITB    ACIACS    WAIT FOR TBE
00741A FC72 27 FB FC6F          BEQ     OUTEE1
00742A FC74 B7 EC01    A        STAA    ACIAXR    SEND THE CHRACTER
00743A FC77 33                  PULB              RESTORE B
00744A FC78 39                  RTS
00745                        *
00746                        * DETERMINE BAUD RATE OF CRT TERMINAL
00747                        *
00748A FC79 B6 E400    A BAUDRT LDAA    LEDDIG    CLEAR IRQ FLAG
00749A FC7C 86 06      A        LDAA    #$06      SET UP CA1 FOR RIZING EDGE
00750A FC7E B7 E401    A        STAA    DIGCTL
00751A FC81 B6 E401    A LOOP   LDAA    DIGCTL    TIME THE BAUD RATE
00752A FC84 08                  INX
00753A FC85 49                  ROLA
00754A FC86 24 F9 FC81          BCC     LOOP
00755A FC88 DF 4A      A        STX     DIGIT1    SAVE THE RATE COUNTER
00756A FC8A CE FEF8    A        LDX     #BDTBL-3
00757A FC8D 08           NXT    INX
00758A FC8E 08                  INX
00759A FC8F 08                  INX
00760A FC90 8C FF16    A        CPX     #EBDTBL   END OF BAUD RATE TABLE
  761A FC93 27 1D FCB2          BEQ     JNITBD
00762A FC95 A6 00      A        LDAA    X         (HIGH BYTE) RATE COUNT
00763A FC97 E6 01      A        LDAB    1,X       (LOW BYTE) RATE COUNT
00764A FC99 91 4A      A        CMPA    DIGIT1
00765A FC9B 25 06 FCA3          BCS     SETBD
00766A FC9D 26 EE FC8D          BNE     NXT
00767A FC9F D1 4B      A        CMPB    DIGIT2
00768A FCA1 22 EA FC8D          BHI     NXT
00769A FCA3 A6 02      A SETBD  LDAA    2,X       GET BAUD CODE FROM TABLE
00770A FCA5 97 43      A        STAA    BAUDCD    SET BAUD RATE
00771A FCA7 BD FCB7    A        JSR     BAUD
00772A FCAA BD FCD4    A        JSR     OUTCRL
00773A FCAD 86 80      A        LDAA    #$80
00774A FCAF 97 5C      A        STAA    IOFLAG    SET FLAG FOR CRT
00775A FCB1 39                  RTS
00776A FCB2 31          JNITBD  INS               CLEAN UP STACK
00777A FCB3 31                  INS
00778A FCB4 7E F8B7    A        JMP     INITBD    LOOK FOR KEY AGAIN

00780                        *
00781                        *  INITIALIZE ACIA AND SET BAUD RATE
00782                        *
00783A FCB7 86 03      A BAUD   LDAA    #3        RESET
00784A FCB9 B7 EC00    A        STAA    ACIACS
00785A FCBC 86 49      A        LDAA    #$49      7 DATA,EVEN PARITY,2 STOP
  786A FCBE B7 EC00    A        STAA    ACIACS
  787A FCC1 7F E400    A        CLR     LEDDIG    DESELECT DISPLAY
00788A FCC4 86 34      A        LDAA    #$34      SELECT BAUD GENERATOR
00789A FCC6 B7 E403    A        STAA    SEGCTL
```

```
00790A FCC9 96 43    A         LDAA   BAUDCD   SELECT BAUD
00791A FCCB B7 E402  A         STAA   LEDSEG
 0792A FCCE 86 3C    A         LDAA   #$3C     RELEASE BAUD GENERATOR
  793A FCD0 B7 E403  A         STAA   SEGCTL
00794A FCD3 39         BAUDX   RTS
00795                  *
00796                  *   CARRIAGE RETURN/LINE FEED AND NULLS
00797                  *
00798A FCD4 CE FFE9  A OUTCRL  LDX    #CRLSTR  CARRIAGE RETURN / LINE FEED
00799                  *
00800                  * PRINT CHARACTER STRING
00801                  *
00802A FCD7 A6 00    A PDATAL  LDAA   X        GET CHARACTER
00803A FCD9 81 04    A         CMPA   #4       END ?
00804A FCDB 27 F6 FCD3         BEQ    BAUDX
00805A FCDD BD FC6C  A         JSR    OUTEE
00806A FCE0 08                 INX
00807A FCE1 20 F4 FCD7         BRA    PDATAL
00808                  *
00809                  * INPUT STRING OF DATA FROM TAPE
00810                  * THIS ROUTINE IS RESPONSIBLE FOR REMOVING THE
00811                  * SECOND $1B'S THAT ARE AUTOMATICALLY INSERTED BY THE
00812                  * TAPE CONTROLLER
00813                  * ENTRY: X->DESTINATION OF DATA
00814                  *        COUNTR=#BYTES TO INPUT
00815                  *
00816                  * EXIT: CARRY CLEAR FOR VALID, CARRY SET FOR INVALID

00818A FCE3 DF 50    A INTAP   STX    DATPTR   SAVE DATA POINTER
  819A FCE5 5F         INTAP0  CLRB            CLEAR ESCAPE COUNTER
00820A FCE6 BD FC44  A INTAP1  JSR    INEE     GET DATA FROM TAPE
00821A FCE9 5D                 TSTB            IF ALREADY 1, BETTER BE 2
00822A FCEA 27 07 FCF3         BEQ    INTAP2
00823A FCEC 81 1B    A         CMPA   #$1B     BETTER BE SECOND ESCAPE
00824A FCEE 27 F5 FCE5         BEQ    INTAP0   IT IS, SO IGNORE SECOND ONE
00825A FCF0 0D                 SEC             SET CARRY AND ABORT
00826A FCF1 20 18 FD0B         BRA    INTAP4
00827A FCF3 81 1B    A INTAP2  CMPA   #$1B     CHECK FOR FIRST ESCAPE
00828A FCF5 26 01 FCF8         BNE    INTAP3
00829A FCF7 5C                 INCB            ESCCNT = 1
00830A FCF8 DE 50    A INTAP3  LDX    DATPTR
00831A FCFA A7 00    A         STAA   0,X      PUT DATA IN MEMORY
00832A FCFC 9B 3D    A         ADDA   CHKSUM
00833A FCFE 97 3D    A         STAA   CHKSUM
00834A FD00 08                 INX
00835A FD01 DF 50    A         STX    DATPTR
00836A FD03 DE 52    A         LDX    COUNTR   COUNT THE DATA BYTE
00837A FD05 09                 DEX
00838A FD06 DF 52    A ,       STX    COUNTR
00839A FD08 26 DC FCE6         BNE    INTAP1
00840A FD0A 0C                 CLC             CLEAR CARRY FOR VALID
00841A FD0B 39         INTAP4  RTS
00842                  *
00843                  * THIS ROUTINE OUTPUTS A STRING OF DATA
0  44                  * TO THE TAPE
00845                  * ENTRY: X->DATA TO OUTPUT
00846                  *        COUNTR=#BYTES TO OUTPUT
```

```
00848A FD0C DF 50      A OUTTAP STX    DATPTR   SAVE DATE POINTER
00849A FD0E DE 50      A OUTTA1 LDX    DATPTR
00850A FD10 A6 00      A        LDAA   0,X      GET DATA TO OUTPUT
00851A FD12 08                  INX
00852A FD13 DF 50      A        STX    DATPTR   POINT TO NEXT
00853A FD15 BD FC6C    A        JSR    OUTEE    SEND THE DATA—
00854A FD18 9B 3D      A        ADDA   CHKSUM   UPDATE CHECKSUM
00855A FD1A 97 3D      A        STAA   CHKSUM
00856A FD1C DE 52      A        LDX    COUNTR   COUNT IT
00857A FD1E 09                  DEX
00858A FD1F DF 52      A        STX    COUNTR
00859A FD21 26 EB FD0E          BNE    OUTTA1   IF NOT DONE CONTINUE
00860A FD23 39                  RTS
00861                  *
00862                  *SET THE ACIA FOR 8-BIT DATA, 9600 BAUD
00863                  *
00864A FD24 96 43      A SACIA  LDAA   BAUDCD   SAVES MONITOR BAUD RATE
00865A FD26 97 42      A        STAA   SBAUD
00866A FD28 86 08      A        LDAA   #B9600   9600 BAUD
00867A FD2A 97 43      A        STAA   BAUDCD   SET RATE FOR 9600
00868A FD2C BD FCB7    A        JSR    BAUD
00869A FD2F 86 19      A        LDAA   #$19     8-BITS DATA
00870A FD31 B7 EC00    A        STAA   ACIACS
00871A FD34 39                  RTS
00872                  *
00873                  *RESTORE BAUD RATE FOR MONITOR ROUTINE
00874                  *
00875A FD35 96 42      A RACIA  LDAA   SBAUD
00876A FD37 97 43      A        STAA   BAUDCD
00877A FD39 7E FCB7    A        JMP    BAUD     RESET BAUD RATE
00878                  *
00879                  * REQUEST AND INPUT ADDRESS
00880                  *
00881                  * ENTRY: B = MESSAGE NUMBER
00882                  * EXIT:  X AND ADDRSS = ADDRESS
00883                  *
00884A FD3C BD FE89    A RQSADD JSR    PUT2     PRINT MESSAGE
00885A FD3F 8D 06 FD47          BSR    GETADD   INPUT ADDRESS
00886A FD41 BD FEF2    A        JSR    DOCRLF   DO CR/LF
00887A FD44 DE 38      A        LDX    ADDRSS   RETURN WITH ADDRESS
00888A FD46 39                  RTS
```

```
00890                      *
00891                      * PARALLEL I/O ROUTINES
00892                      *
00893A FD47 C6 05     A GETADD LDAB    #5        UNDERLINES IN DIGITS 1 - 4
00894A FD49 BD FE9E   A        JSR     PUT4
00895A FD4C BD FDEF   A        JSR     SETDPT    SET DIGIT POINTER
00896A FD4F BD FD7C   A        JSR     IN2HEX    GET HIGH BYTE OF ADDRESS
00897A FD52 97 38     A        STAA    ADDRSS    SAVE HIGH BYTE
00898A FD54 BD FD7C   A        JSR     IN2HEX    GET LOW BYTE OF ADDRESS
00899A FD57 97 39     A        STAA    ADDRSS+1 SAVE LOW BYTE
00900A FD59 BD FEEB   A        JSR     RQSSPC    REQUEST A SPACE
00901A FD5C 7E FECD   A        JMP     CHKIOP

00903A FD5F BD FDEF   A ADROUT JSR     SETDPT    SET DIGIT POINTER
00904A FD62 96 38     A        LDAA    ADDRSS    GET UPPER 2 ADR DIGITS
00905A FD64 8D 05 FD6B         BSR     OUT2HX    OUTPUT 2 HEX DIGITS
00906A FD66 BD FEEB   A        JSR     RQSSPC    REQUEST A SPACE
00907A FD69 96 39     A        LDAA    ADDRSS+1 GET LOWER 2 DIGITS

00909A FD6B 16          OUT2HX TAB               PUT 2 HEX DIGITS IN B ALSO
00910A FD6C 44                 LSRA              GET UPPER
00911A FD6D 44                 LSRA              HEX DIGIT
00912A FD6E 44                 LSRA              INTO LOWER
00913A FD6F 44                 LSRA              HALF OF WORD
00914A FD70 BD FE66   A        JSR     WRDATA    WRITE IT TO NEXT DIGIT CELL
00915A FD73 17                 TBA               GET LOWER HEX DIGIT
00916A FD74 84 0F     A        ANDA    #15       INTO LOWER HALF OF WORD
00917A FD76 BD FE66   A        JSR     WRDATA    WRITE IT TO NEXT DIGIT CELL
00918A FD79 7E FECD   A        JMP     CHKIOP    EXIT - CHECK I/O FLAG

00920A FD7C BD FE55   A IN2HEX JSR     INPUT     GET A HEX DIGIT
00921A FD7F 48                 ASLA              PUT IT
00922A FD80 48                 ASLA              IN THE
00923A FD81 48                 ASLA              UPPER HALF
00924A FD82 48                 ASLA              OF THE WORD
00925A FD83 36                 PSHA              SAVE IT
00926A FD84 BD FE55   A        JSR     INPUT     GET SECOND HEX DIGIT
00927A FD87 33                 PULB              GET FIRST DIGIT BACK
00928A FD88 1B                 ABA               COMBINE THE TWO DIGITS
00929A FD89 39                 RTS               EXIT
00930                      *
00931                      *  WATCH KEYBOARD
00932                      *
00933A FD8A D6 45     A FIND   LDAB    FOUND     IF A PUSH BUTTON WAS ALSO
00934A FD8C 26 08 FD96         BNE     KEYIN     FOUND LAST SCAN SKIP DATA WRITE
00935A FD8E 97 47     A        STAA    INCHAR
00936A FD90 BD FE66   A        JSR     WRDATA    ELSE WRITE THE DIGIT
00937A FD93 7C 0045   A        INC     FOUND     AND SET FOUND FLAG
00938A FD96 8D 20 FDB3 KEYIN   BSR     DISPLY    DISPLAY ALL LED DIGITS
00939A FD98 86 0F     A        LDAA    #15       INIT KEY NUMBER
00940A FD9A F6 E002   A        LDAB    KBSTOP    GET 8 BITS OF KB
00941A FD9D 58          UPLOOP ASLB              LOOK AT NEXT BIT
00942A FD9E 24 EA FD8A         BCC     FIND      IF IT IS ZERO GOTO FIND
00943A FDA0 4A                 DECA              BUMP NUMBER
 0944A FDA1 81 07     A        CMPA    #7        IN POT 7 YET
00945A FDA3 26 F8 FD9D         BNE     UPLOOP    GO BACK
00946A FDA5 F6 E000   A        LDAB    KBOT07    GET OTHER 8 BITS OF KB
00947A FDA8 58          LOLOOP ASLB              LOOK AT NEX BIT
```

```
00948A FDA9 24 DF FD8A          BCC    FIND      IF IT IS ZERO GOTO FIND
00949A FDAB 4A                  DECA             BUMP NUMBER
00950A FDAC 2C FA FDA8          BGE    LOLOOP    GO BACK
00951A FDAE 96 45      A        LDAA   FOUND     GOT A KEY
00952A FDB0 27 E4 FD96          BEQ    KEYIN     NO - LOOK AGAIN
00953A FDB2 7F 0045    A        CLR    FOUND     RESET FOUND FLAG
00954A FDB5 96 47      A        LDAA   INCHAR    CHARACTER IN A
00955A FDB7 39                  RTS
00956                    *
00957                    *  DISPLAY SIX CHARACTERS
00958                    *
00959A FDB8 DF 2F      A DISPLY STX    SAVEX     SAVE X, A, AND B
00960A FDBA 36                  PSHA
00961A FDBB 37                  PSHB
00962A FDBC CE 004F    A        LDX    #DIGIT6   POINT TO LAST DIGIT
00963A FDBF 86 01      A        LDAA   #1        SELECT DIGIT 7
00964A FDC1 5F           DGLOOP CLRB             CLEAR
00965A FDC2 F7 E402    A        STAB   LEDSEG    AND
00966A FDC5 48                  ASLA             SELECT NEXT DIGIT
00967A FDC6 B7 E400    A        STAA   LEDDIG    STROBE IT
00968A FDC9 2A 05 FDD0          BPL    NOEXIT    IF DONE 6 TIMES
00969A FDCB 33                  PULB             RESTORE X, A, AND B
00970A FDCC 32                  PULA
00971A FDCD DE 2F      A        LDX    SAVEX
00972A FDCF 39                  RTS              EXIT
00973A FDD0 E6 00      A NOEXIT LDAB   0,X       GET 7-SEG DISPLAY CODE
00974A FDD2 F7 E402    A        STAB   LEDSEG    WRITE IT OUT
00975A FDD5 5F                  CLRB             DELAY FOR
00976A FDD6 5C           DELAY  INCB             1538 MPU
00977A FDD7 26 FD FDD6          BNE    DELAY     CYCLES
00978A FDD9 09                  DEX              POINT TO NEXT DIGIT
00979A FDDA 20 E5 FDC1          BRA    DGLOOP    GO BACK FOR NEXT DIGIT
```

```
00981                      *
00982                      * GENERAL UTILITY ROUTINES
00983                      *


00985                      *
00986                      *  ADD A AND B TO X
00987                      *
00988A FDDC DF 2C     A ADDABX STX    TEMPX     TEMP STORE FOR X
00989A FDDE DB 2D     A        ADDB   TEMPX+1   ADD LOW BYTE
00990A FDE0 99 2C     A        ADCA   TEMPX     ADD HIGH BYTE
00991A FDE2 97 2C     A        STAA   TEMPX
00992A FDE4 D7 2D     A        STAB   TEMPX+1
00993A FDE6 DE 2C     A        LDX    TEMPX     GET NEW X
00994A FDE8 39                 RTS
00995                      *
00996                      *  ADD B ONLY TO X
00997                      *
00998A FDE9 36           ADDBX  PSHA             SAVE A
00999A FDEA 4F                  CLRA             CLEAR MSB
01000A FDEB 8D EF FDDC          BSR    ADDABX
01001A FDED 32                  PULA             RESTORE A
01002A FDEE 39                  RTS
01003                      *
01004                      * SET DIGIT POINTER
01005                      *
01006A FDEF CE 004A   A SETDPT LDX    #DIGIT1   POINT TO FIRST
01007A FDF2 DF 48     A        STX    DIGPTR
01008A FDF4 39                 RTS
1009                       *
01010                      * MOVE PROM INTO RAM
01011                      *
01012                      * FIRST PROM MUST BE AT $9000
01013                      *
01014                      * LOCATIONS $9000-$900F ARE RESERVED
01015                      * FOR A BANNER MESSAGE DISPLAYED BY PDATAL
01016                      *
01017            9000   A PROM   EQU    $9000     PROM BANNER BEGINNING
01018                      *
01019                      * LOCATIONS $9010-$901B CONTAIN CONTROL
01020                      * DATA FOR THE ACTUAL MOVE
01021                      *
01022            9010   A SRCBEG EQU    $9010     SOURCE BEGINNING ADDRESS
01023            9012   A SRCEND EQU    $9012     SOURCE END ADDRESS
01024            9014   A DSTBEG EQU    $9014     DESTINATION BEGINNING ADDRESS
01025            9016   A DSTEND EQU    $9016     DESTINATION END ADDRESS
01026            9018   A DSTSTK EQU    $9018     DESTINATION STACK ADDRESS
01027            901A   A DSTGOV EQU    $901A     DESTINATION START VECTOR
01028                      *
01029A FDF5 7D 005C   A LPROM  TST    IOFLAG    CRT ACTIVE?
01030A FDF8 2A 2D FE27         BPL    NOCRT     NO
01031A FDFA FE 9000   A        LDX    PROM      CHECK FOR VALID PROM
01032A FDFD 8C 4153   A        CPX    #$4153    IS IT 'AS'?
01033A FE00 26 38 FE3A         BNE    NOPROM    NO
 1034A FE02 BD FCD4   A        JSR    OUTCRL    SEND CR/LF TO TUBE
 1035A FE05 CE 9000   A        LDX    #PROM     SEND BANNER
01036A FE08 BD FCD7   A        JSR    PDATAL
01037A FE0B BD FCD4   A        JSR    OUTCRL    SEND CR/LF TO CRT
```

```
01038A FE0E BE 9016   A         LDS    DSTEND    POINT TO DESTINATION
01039A FE11 FE 9012   A         LDX    SRCEND    POINT TO SOURCE
01040A FE14 A6 00     A LPROM1  LDAA   0,X       GET BYTE
01041A FE16 36                  PSHA             MOVE IT
01042A FE17 BC 9010   A         CPX    SRCBEG    DONE?
01043A FE1A 27 03 FE1F          BEQ    LPROM2    YES
01044A FE1C 09                  DEX              NO
01045A FE1D 20 F5 FE14          BRA    LPROM1    MOVE ANOTHER BYTE
01046A FE1F BE 9018   A LPROM2  LDS    DSTSTK    GET TARGET STACK POINTER
01047A FE22 FE 901A   A         LDX    DSTGOV    GET START VECTOR
01048A FE25 6E 00     A         JMP    0,X       GO!
01049A FE27 C6 06     A NOCRT   LDAB   #$6
01050A FE29 BD FE9E   A         JSR    PUT4      "ACIA" => DISPLAY
01051A FE2C C6 02     A         LDAB   #$2
01052A FE2E BD FEE8   A         JSR    RQSCRL
01053A FE31 BD FE89   A         JSR    PUT2
01054A FE34 8E 0026   A LOUT    LDS    #TOPSTK   RESTORE BUG STACK
01055A FE37 7E F8E9   A         JMP    CONTRL
01056A FE3A BD FCD4   A NOPROM  JSR    OUTCRL
01057A FE3D CE FFD9   A         LDX    #NPROM
01058A FE40 BD FCD7   A         JSR    PDATAL    PRINT "CAN'T FIND BASIC"
01059A FE43 BD FCD4   A         JSR    OUTCRL
01060A FE46 20 EC FE34          BRA    LOUT      BACK TO CONTROL
01061                   *
01062                   * PROVIDE DELAY FOR TAPE TO START
01063                   *
01064A FE48 86 78     A ONDELA  LDAA   #120      12 SECONDS
01065A FE4A CE 2710   A ALOOP   LDX    #10000    100 MILLISEC LOOP
01066A FE4D 09          XLOOP   DEX              10 MICROSEC HERE
01067A FE4E 01                  NOP              TO
01068A FE4F 26 FC FE4D          BNE    XLOOP     HERE
01069A FE51 4A                  DECA
01070A FE52 26 F6 FE4A          BNE    ALOOP
01071A FE54 39                  RTS
```

```
01073                        *
01074                        * GENERAL I/O ROUTINES
01075                        *
01076A FE55 96 5C        A INPUT  LDAA   IOFLAG
01077A FE57 2B 03 FE5C          BMI    *+5
01078A FE59 7E FD96   A         JMP    KEYIN
01079A FE5C BD FC31   A         JSR    INHEX
01080A FE5F 97 47     A INCRTO STAA   INCHAR
01081A FE61 8D 03 FE66          BSR    WRDATA   WRITE THE CHARACTER
01082A FE63 96 47     A         LDAA   INCHAR   RESTORE THE CHAR TO A
01083A FE65 39                  RTS
01084                        *
01085                        *  CONVERT HEX DIGIT TO LED SEGMENT CODE
01086                        *
01087A FE66 37              WRDATA PSHB            SAVE B
01088A FE67 16                     TAB             PUT HEX DIGIT IN B FOR OFFSET
01089A FE68 7D 0044   A            TST    NOWRF   IF NO-WRITE FLAG IS SET
01090A FE6B 26 17 FE84             BNE    WRDONE  SKIP THE WRITE
01091A FE6D 7D 005C   A            TST    IOFLAG  CRT ACTIVE?
01092A FE70 2A 03 FE75             BPL    WR2LED  NO
01093A FE72 BD FC64   A            JSR    OUTHEX  YES
01094A FE75 CE FF51   A WR2LED LDX    #TBL    GET TABLE ADDRESS
01095A FE78 BD FDE9   A            JSR    ADDBX   DETERMINE CHARACTER ADDRESS
01096A FE7B A6 00     A            LDAA   0,X     GET THE DISPLAY CODE
01097A FE7D DE 48     A            LDX    DIGPTR  GET DIGIT POINTER
01098A FE7F A7 00     A            STAA   0,X     DISPLAY THE DIGIT
01099A FE81 08                     INX            POINT TO NEXT
01100A FE82 DF 48     A            STX    DIGPTR  DIGIT
01101A FE84 7F 0044   A WRDONE CLR    NOWRF   CLEAR NO-WRITE FLAG
01102A FE87 33                     PULB           RESTORE B
01103A FE88 39                     RTS
01104                       *
01105                       * DISPLAY TWO CHARACTERS
01106                       *
01107                       * CHECK FOR ACTIVE CRT, AND IF THERE
01108                       * SEND CHARACTERS TO IT, CHARACTERS
01109                       * ALWAYS GO TO LED'S
01110                       *
01111                       * B = DISPLAY CODE ON ENTRY
01112                       *
01113A FE89 58              PUT2   ASLB           CALCULATE OFFSET OF STRING
01114A FE8A 58                     ASLB
01115A FE8B CE FFB1   A            LDX    #STRNG2 POINT TO TABLE BASE
01116A FE8E BD FDE9   A            JSR    ADDBX   GET STRING ADDRESS
01117A FE91 A6 00     A            LDAA   0,X
01118A FE93 97 4E     A            STAA   DIGIT5
01119A FE95 08                     INX
01120A FE96 A6 00     A            LDAA   0,X
01121A FE98 97 4F     A            STAA   DIGIT6
01122A FE9A C6 02     A            LDAB   #2
01123A FE9C 20 1E FEBC             BRA    PUTCRT
01124                       *
01125                       * DISPLAY FOUR CHARACTERS
01126                       *
 1127A FE9E 58              PUT4   ASLB
01128A FE9F 58                     ASLB
01129A FEA0 58                     ASLB
01130A FEA1 CE FF61   A            LDX    #STRNG4
```

```
01131A FEA4 BD FDE9  A        JSR    ADDBX
01132A FEA7 A6 00    A        LDAA   0,X
01133A FEA9 97 4A    A        STAA   DIGIT1
01134A FEAB 08                INX
01135A FEAC A6 00    A        LDAA   0,X
01136A FEAE 97 4B    A        STAA   DIGIT2
01137A FEB0 08                INX
01138A FEB1 A6 00    A        LDAA   0,X
01139A FEB3 97 4C    A        STAA   DIGIT3
01140A FEB5 08                INX
01141A FEB6 A6 00    A        LDAA   0,X
01142A FEB8 97 4D    A        STAA   DIGIT4
01143A FEBA C6 04    A        LDAB   #4
01144A FEBC 96 5C    A PUTCRT LDAA   IOFLAG    CRT ACTIVE?
01145A FEBE 2A 21 FEE1        BPL    PUTRTS    NO - GET OUT
01146A FEC0 8A 01    A        ORAA   #$1       YES - REQUEST SPACE
01147A FEC2 97 5C    A        STAA   IOFLAG
01148A FEC4 08         PUTASC INX              SEND STRING TO CRT
01149A FEC5 A6 00    A        LDAA   0,X
01150A FEC7 BD FC6C  A        JSR    OUTEE
01151A FECA 5A                DECB
01152A FECB 26 F7 FEC4        BNE    PUTASC
01153A FECD D6 5C    A CHKIOF LDAB   IOFLAG
01154A FECF 2A 10 FEE1        BPL    PUTRTS
01155A FED1 C5 01    A        BITB   #1        SPACE REQUEST?
01156A FED3 27 05 FEDA        BEQ    *+7       NO
01157A FED5 86 20    A        LDAA   #$20      YES - SEND SPACE
01158A FED7 BD FC6C  A        JSR    OUTEE
01159A FEDA C5 02    A        BITB   #2        CR/LF REQUEST?
 1160A FEDC 27 03 FEE1        BEQ    *+5       NO
01161A FEDE BD FCD4  A        JSR    OUTCRL
01162A FEE1 D6 5C    A PUTRTS LDAB   IOFLAG    GET I/O FLAG
01163A FEE3 C4 FC    A        ANDB   #$FC      CLEAR THE REQUESTS
01164A FEE5 D7 5C    A        STAB   IOFLAG
01165A FEE7 39                RTS
01166                  *
01167                  * SET REQUEST
01168                  *
01169A FEE8 86 02    A RQSCRL LDAA   #2        REQUEST CR/LF
01170A FEEA    8C    A        FCB    $8C       SKIP 2
01171A FEEB 86 01    A RQSSPC LDAA   #1        REQUEST SPACE
01172A FEED 9A 5C    A        ORAA   IOFLAG
01173A FEEF 97 5C    A        STAA   IOFLAG
01174A FEF1 39                RTS
01175                  *
01176                  * SEND CARRIAGE RETURN / LINE FEED
01177                  * IF CRT ACTIVE
01178                  *
01179A FEF2 37         DOCRLF PSHB             SAVE B
01180A FEF3 36                PSHA             SAVE A
01181A FEF4 8D F2 FEE8        BSR    RQSCRL    SET REQUEST FLAG
01182A FEF6 8D D5 FECD        BSR    CHKIOF    SEND IT
01183A FEF8 32                PULA             RESTORE A
01184A FEF9 33                PULB             RESTORE B
  185A FEFA 39                RTS
```

```
01187                      *
01188                      * BAUD RATE TABLE
01189                      *
01190A FEFB     02       A BDTBL  FCB     $02,$30,$0F 110 BAUD
01191A FEFE     01       A        FCB     $01,$61,$0E 150 BAUD
01192A FF01     00       A        FCB     $00,$AF,$0D 300 BAUD
01193A FF04     00       A        FCB     $00,$56,$06 600 BAUD
01194A FF07     00       A        FCB     $00,$2F,$0B 1200 BAUD
01195A FF0A     00       A        FCB     $00,$20,$0A 1800 BAUD
01196A FF0D     00       A        FCB     $00,$15,$0C 2400 BAUD
01197A FF10     00       A        FCB     $00,$08,$09 4800 BAUD
01198A FF13     00       A        FCB     $00,$00,$08 9600 BAUD
01199          FF16      A EBDTBL EQU     *
```

```
01201                      *
01202                      *  TABLE OF ILLEGAL OP CODES
01203                      *
01204A FF16     00     A VALOP  FCB     0,2,3,4,5
01205A FF1B     12     A        FCB     $12,$13,$14,$15,$18,$1A,$1C,$1D,$1E,$1F
01206A FF25     21     A        FCB     $21
01207A FF26     38     A        FCB     $38,$3A,$3C,$3D
01208A FF2A     41     A        FCB     $41,$42,$45,$4B,$4E
01209A FF2F     51     A        FCB     $51,$52,$55,$5B,$5E
01210A FF34     61     A        FCB     $61,$62,$65,$6B
01211A FF38     71     A        FCB     $71,$72,$75,$7B
01212A FF3C     83     A        FCB     $83,$87,$8F
01213A FF3F     93     A        FCB     $93,$9D
01214A FF41     A3     A        FCB     $A3
01215A FF42     B3     A        FCB     $B3
01216A FF43     C3     A        FCB     $C3,$C7,$CC,$CD,$CF
01217A FF48     D3     A        FCB     $D3,$DC,$DD
01218A FF4B     E3     A        FCB     $E3,$EC,$ED
01219A FF4E     F3     A        FCB     $F3,$FC,$FD
01220           FF51   A EVALOP EQU     *
01221                      *
01222                      *  TABLE OF HEX DIGIT DISPLAY
01223                      *  CODES
01224                      *
01225A FF51     FC     A TBL    FCB     $FC        0
01226A FF52     60     A        FCB     $60        1
01227A FF53     DA     A        FCB     $DA        2
01228A FF54     F2     A        FCB     $F2        3
01229A FF55     66     A        FCB     $66        4
01230A FF56     B6     A        FCB     $B6        5
01231A FF57     BE     A        FCB     $BE        6
01232A FF58     E0     A        FCB     $E0        7
01233A FF59     FE     A        FCB     $FE        8
01234A FF5A     E6     A        FCB     $E6        9
01235A FF5B     EE     A        FCB     $EE        A
01236A FF5C     3F     A        FCB     $3F        B
01237A FF5D     9C     A        FCB     $9C        C
01238A FF5E     7A     A        FCB     $7A        D
01239A FF5F     9E     A        FCB     $9E        E
01240A FF60     8E     A        FCB     $8E        F
01241           FF61   A ENDTBL EQU     *
```

```
01243                        *
01244                        * DISPLAY STRINGS DEFINED
01245                        *
01246                        * FOUR CHARACTER STRINGS
01247                        *
01248A FF61    7AFC   A STRNG4 FDB    $7AFC    DONE : CODE = 0   00
01249A FF63    2A9E   A        FDB    $2A9E
01250A FF65    44     A        FCC    4,DONE
01251                        *
01252A FF69    1CFC   A        FDB    $1CFC    LOAD : CODE = 1
01253A FF6B    EE7A   A        FDB    $EE7A
01254A FF6D    4C     A        FCC    4,LOAD
01255                        *
01256A FF71    3EEE   A        FDB    $3EEE    BAD- : CODE = 2
01257A FF73    7A02   A        FDB    $7A02
01258A FF75    42     A        FCC    4,BAD-
01259                        *
01260A FF79    BE3A   A        FDB    $BE3A    GOOD : CODE = 3
01261A FF7B    3A7A   A        FDB    $3A7A
01262A FF7D    47     A        FCC    4,GOOD
01263                        *
01264A FF81    EEB6   A        FDB    $EEB6    ASCI : CODE = 4
01265A FF83    9C0C   A        FDB    $9C0C
01266A FF85    41     A        FCC    4,ASCI
01267                        *
01268A FF89    1010   A        FDB    $1010    UNDERLINES : CODE = 5
01269A FF8B    1010   A        FDB    $1010
01270A FF8D    0000   A        FDB    0,0
01271                        *
01272A FF91    EE9C   A        FDB    $EE9C    ACIA : CODE = 6
01273A FF93    0CEE   A        FDB    $0CEE
01274A FF95    41     A        FCC    4,ACIA
01275                        *
01276A FF99    0000   A        FDB    $0000    PCCC : CODE 7
01277A FF9B    0000   A        FDB    $0000
01278A FF9D    50     A        FCC    4,PCCC
01279                        *
01280A FFA1    0000   A        FDB    $0000    X A : CODE 8
01281A FFA3    0000   A        FDB    $0000
01282A FFA5    58     A        FCC    4,X A
01283                        *
01284A FFA9    0000   A        FDB    $0000    SP B : CODE 9   09
01285A FFAB    0000   A        FDB    $0000
01286A FFAD    53     A        FCC    4,SP B
```

LDAB# code
JSR PUT4

```
01288                      *
01289                      * TWO CHARACTER STRINGS
01290                      *
01291A FFB1    1010   A STRNG2 FDB     $1010     UNDERLINES : CODE = 0
01292A FFB3    0000   A        FDB     0
01293                      *
01294A FFB5    0000   A        FDB     $0000     BLANKS : CODE = 1
01295A FFB7    0000   A        FDB     0
01296                      *
01297A FFB9    CACA   A        FDB     $CACA     ?? : CODE = 2
01298A FFBB    3F     A        FCC     2,??
01299                      *
01300A FFBD    0202   A        FDB     $0202     HYPHEN : CODE = 3
01301A FFBF    2D     A        FCC     2,--
01302                      *
01303A FFC1    3FCE   A        FDB     $3FCE     BP : CODE = 4
01304A FFC3    42     A        FCC     2,BP
01305                      *
01306A FFC5    3E9E   A        FDB     $3E9E     BE : CODE = 5
01307A FFC7    42     A        FCC     2,BE
01308                      *
01309A FFC9    9E2A   A        FDB     $9E2A     EN : CODE = 6
01310A FFCB    45     A        FCC     2,EN
01311                      *
01312A FFCD    7CCE   A        FDB     $7CCE     UP : CODE = 7
01313A FFCF    55     A        FCC     2,UP
01314                      *
01315A FFD1    CE9C   A        FDB     $CE9C     PC : CODE = 8
01316A FFD3    50     A        FCC     2,PC
01317                      *
01318A FFD5    0000   A        FDB     $0000     BLANKS : CODE = 9
01319A FFD7    4D     A        FCC     2,M-      EXAMINE PROMPT
```

LDAB #code
JSR PUTZ

```
01321                        *
01322                        * MISCELLANEOUS STRINGS
01323                        *
 1324                        *
01325A FFD9     43     A NPROM  FCC     /CAN'T FIND PROM/
01326A FFE8     04     A        FCB     4
01327                        *
01328                        * CARRIAGE RETURN, LINE FEED
01329                        *
01330A FFE9     0D     A CRLSTR FCB     $0D,$0A,0,0,0,4


01332                        *
01333                        *  SET THE VECTORS
01334                        *
01335A FFF8                         ORG     $FFF8
01336A FFF8     F870   A            FDB     IRQ       MASKABLE INT HANDLER ADR
01337A FFFA     FBC5   A            FDB     SWISRV    SOFTWARE INT HANDLER ADR
01338A FFFC     F86C   A            FDB     NMI       NON MASKABLE INT HANDLE ADR
01339A FFFE     F874   A            FDB     RSTART    RESET HANDLER ADR
01340                               END
TOTAL ERRORS 00000--00000
```